

Software-Projekt I

Prof. Dr. Rainer Koschke

Arbeitsgruppe Softwaretechnik
Fachbereich Mathematik und Informatik
Universität Bremen

Sommersemester 2013

Software-Architektur I

1 Software-Architektur

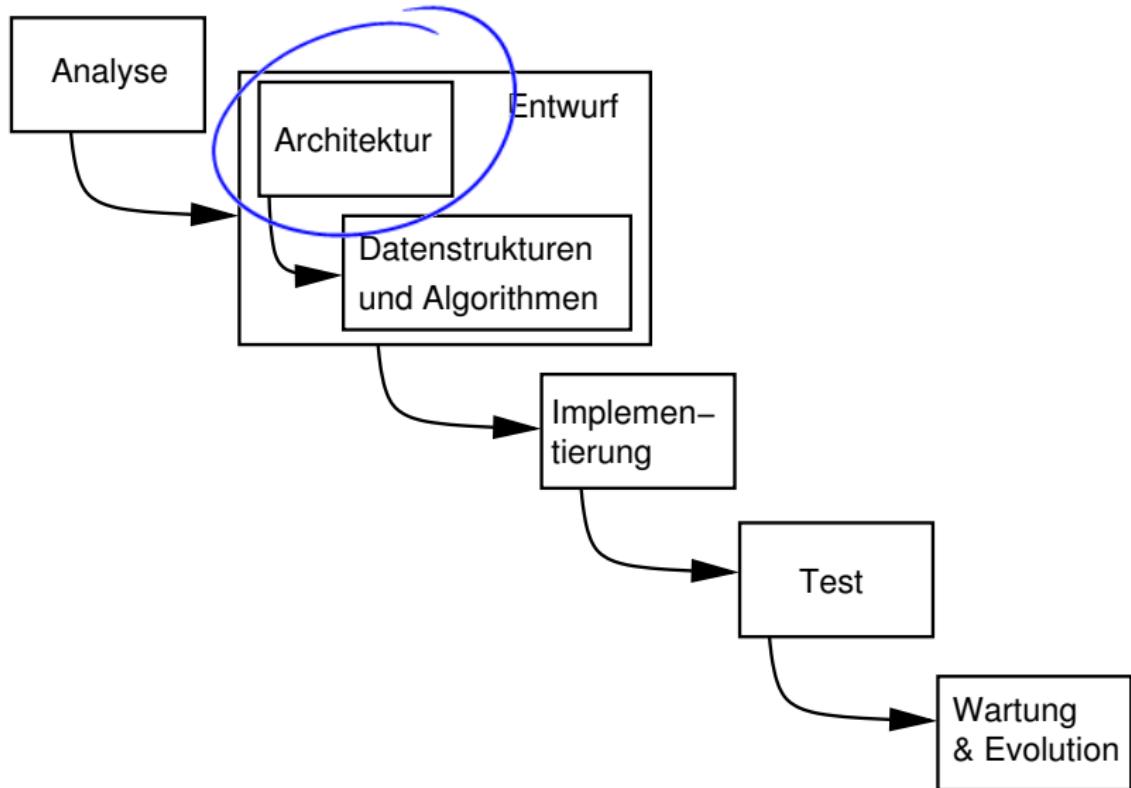
- Was ist Software-Architektur?
- Einflussfaktoren
- Architektsichten und -blickwinkel
- Konzeptioneller Blickwinkel
- Modulblickwinkel
- Ausführungsblickwinkel
- Code-Sicht
- Modularisierung
- Kopplung und Zusammenhalt
- Qualitäten
- Bewertung von Architekturen

Fragen

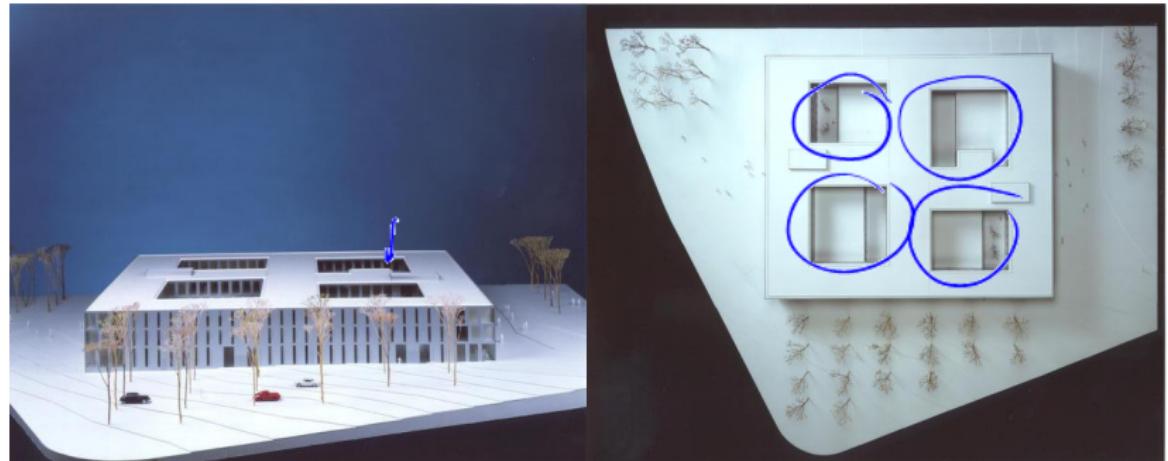


- Was ist Software-Architektur?
- Welche Bedeutung kommt ihr zu?

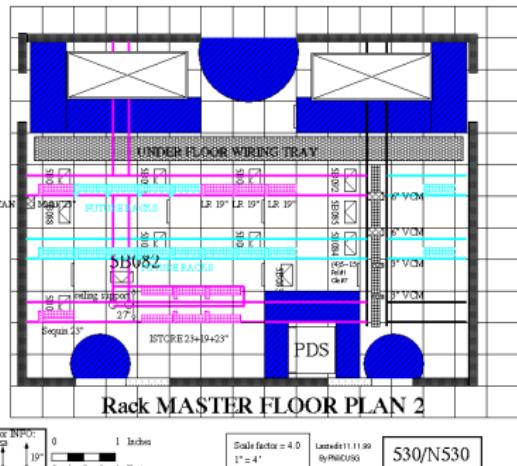
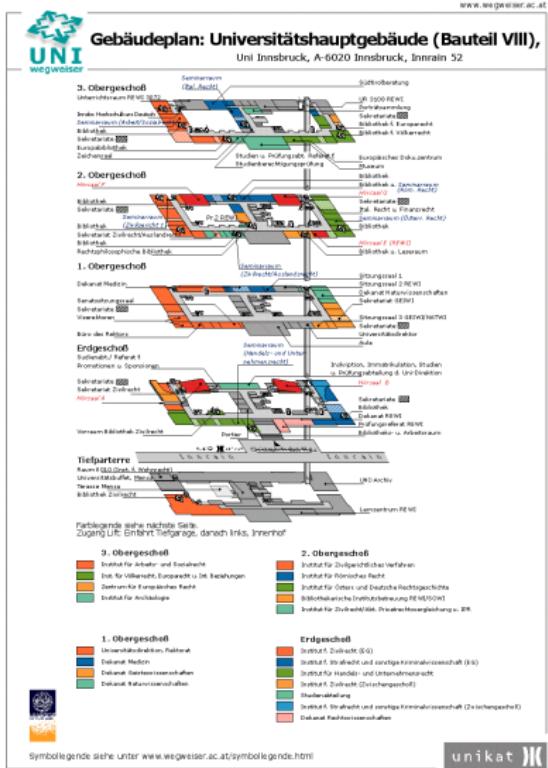
Kontext



Gebäudearchitektur



Gebäudearchitektur



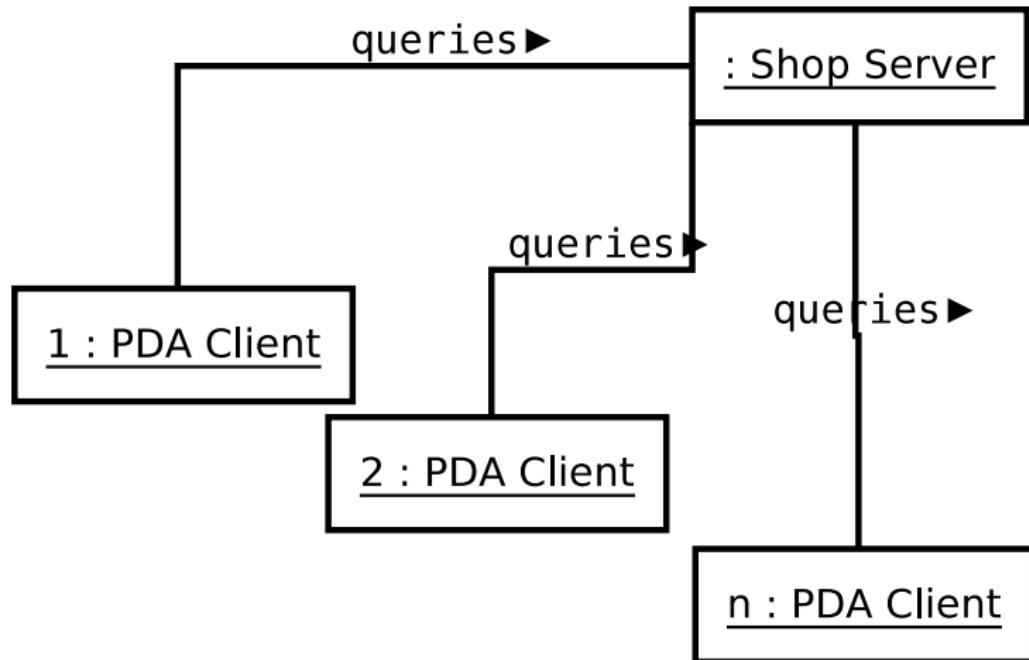
Das fertige Gebäude



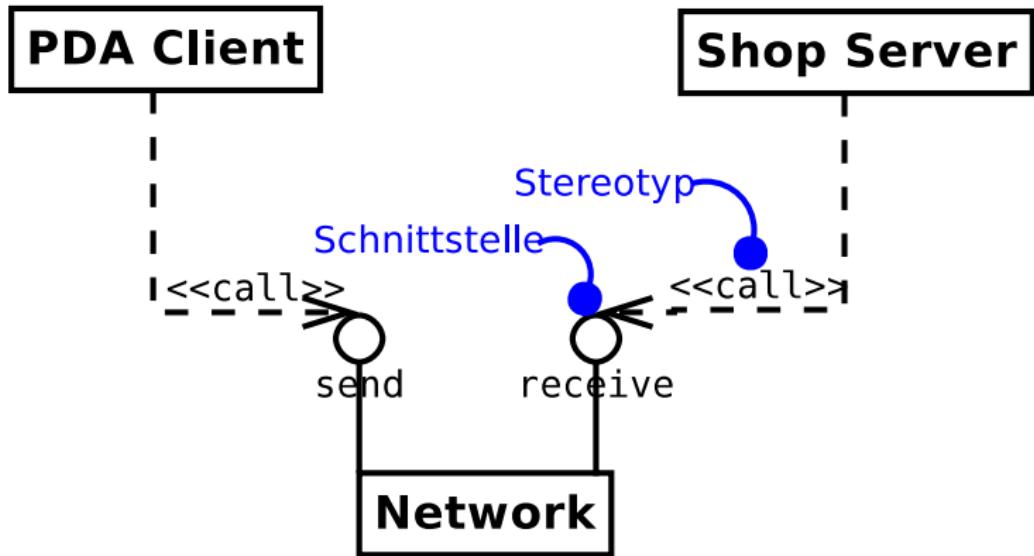
Gebäudearchitektur in der Wartung



Client-Server-Architektur: Dynamische Sicht



Client-Server-Architektur: Statische Sicht



Was ist Architektur?

Architecture is the human organization of empty space using raw material.

Richard Hooker, 1996.

Was ist Architektur?

Architecture is the human organization of empty space using raw material.

Richard Hooker, 1996.

Definition

Software-Architektur ist die grundlegende Organisation eines Systems verkörpert (IEEE P1471 2002)

- in seinen Komponenten,
- deren Beziehungen untereinander und zu der Umgebung
- und die Prinzipien, die den Entwurf und die Evolution leiten.

Bedeutung von Software-Architektur

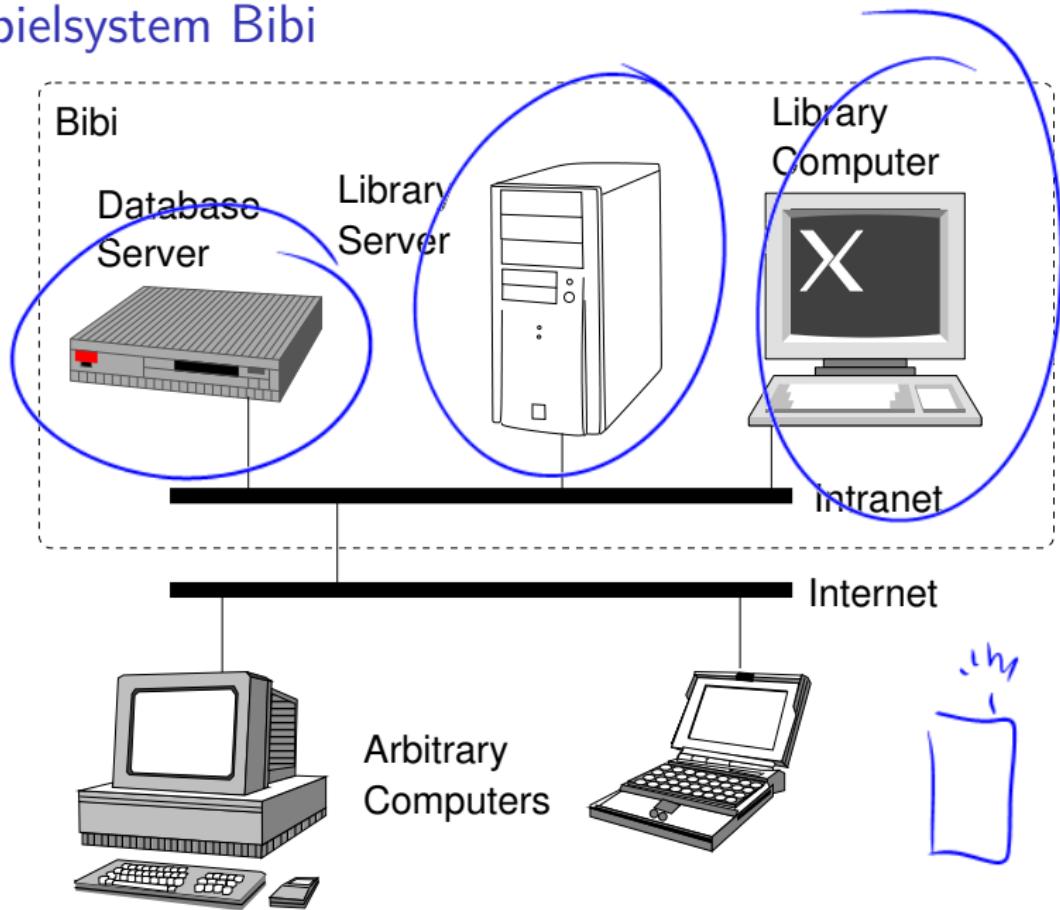
- Kommunikation zwischen allen Interessenten
 - ▶ Aufteilung der Arbeit in unabhängig bearbeitbare Teile
 - ▶ hoher Abstraktionsgrad, der von vielen verstanden werden kann
- Frühe Entwurfsentscheidungen
 - nachhaltige Auswirkungen
 - frühzeitige Analyse
- Transferierbare Abstraktion des Systems
 - eigenständig wiederverwendbar
 - unterstützt Wiederverwendung im großen Stil (Software-Produktlinien)

Fragen



Wie entwirft man eine angemessene Software-Architektur?

Beispielsystem Bibi



Einflussfaktoren/Anliegen (Concerns)

Einflussfaktoren

- Produktfunktionen und -attribute
 - ▶ z.B. Performanz, Ansprüche an Zuverlässigkeit, Umfang und Stabilität der Anforderungen
- Organisation
 - ▶ z.B. Budget, verfügbare Zeit, Team-Größe und -erfahrung
- Technologie
 - ▶ z.B. verfügbare Hard- und Software

Keiner der Faktoren kann isoliert behandelt werden → globale Analyse.

Einflussfaktoren

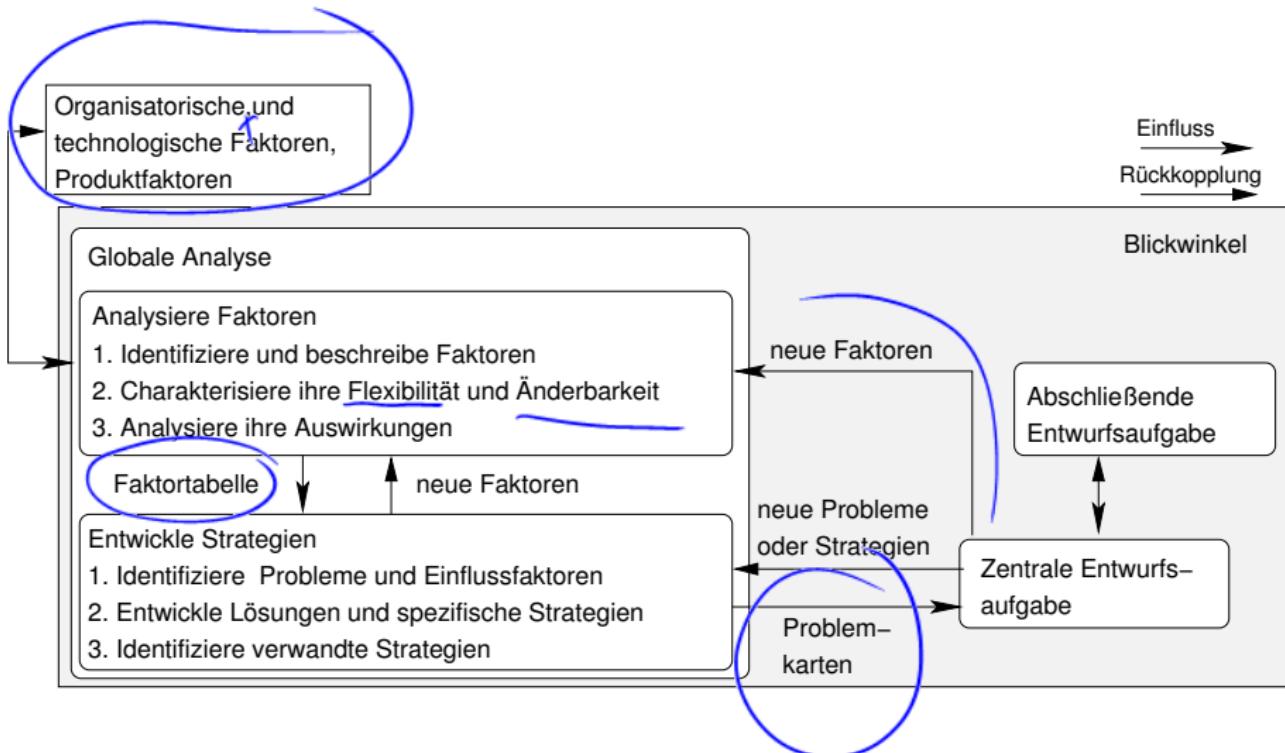
Anmerkung

Ein Faktor ist nur dann relevant, wenn er Einfluss auf die Architektur hat.
Test:

- Architektur A1 ergibt sich mit Betrachtung des Faktors
- Architektur A2 ergibt sich ohne Betrachtung des Faktors

Nur wenn A1 und A2 sich unterscheiden, ist der Faktor relevant.

Globale Analyse nach Hofmeister u. a. (2000)



Globale Analyse: Analysiere Faktoren

1. Identifizierte und beschreibe Faktoren

- Faktoren, die viele Komponenten betreffen
- Faktoren, die sich über die Zeit ändern
- Faktoren, die schwer zu erfüllen sind
- Faktoren, mit denen man wenig Erfahrung hat

Beispiele:

- *Umfang funktionaler Anforderungen*
- *Abgabetermin*
- *Stabilität der Hardware-Plattform*

Globale Analyse: Analysiere Faktoren

2.a Charakterisiere ihre Flexibilität

Flexibilität

Kann zu Beginn mit Beteiligten (Manager, Marketingleute, Kunde, Benutzer etc.) über Faktoren verhandelt werden?

Globale Analyse: Analysiere Faktoren

2.a Charakterisiere ihre Flexibilität

Flexibilität

Kann zu Beginn mit Beteiligten (Manager, Marketingleute, Kunde, Benutzer etc.) über Faktoren verhandelt werden?

Relevante Fragen zur Flexibilität (am Beispiel *Auslieferung der Funktionalität*):

- Kann der Faktor beeinflusst oder geändert werden, so dass der Architekturentwurf vereinfacht werden kann?
 - *Auslieferung der Funktionalität ist verhandelbar. Weniger wichtige Funktionalität kann in späterem Release ausgeliefert werden.*

Globale Analyse: Analysiere Faktoren

2.a Charakterisiere ihre Flexibilität

Flexibilität

Kann zu Beginn mit Beteiligten (Manager, Marketingleute, Kunde, Benutzer etc.) über Faktoren verhandelt werden?

Relevante Fragen zur Flexibilität (am Beispiel *Auslieferung der Funktionalität*):

- Kann der Faktor beeinflusst oder geändert werden, so dass der Architekturentwurf vereinfacht werden kann?
 - ▶ *Auslieferung der Funktionalität ist verhandelbar. Weniger wichtige Funktionalität kann in späterem Release ausgeliefert werden.*
- Wie kann man ihn beeinflussen?
 - ▶ *Nur nach rechtzeitiger Absprache mit dem Kunden.*

Globale Analyse: Analysiere Faktoren

2.a Charakterisiere ihre Flexibilität

Flexibilität

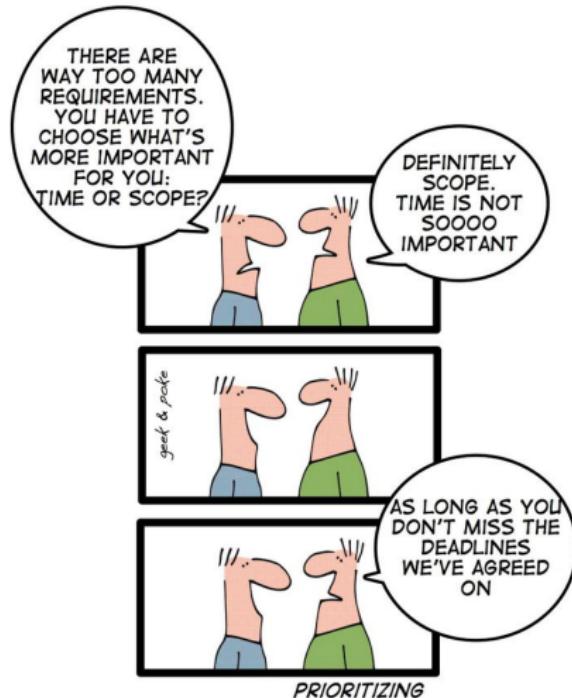
Kann zu Beginn mit Beteiligten (Manager, Marketingleute, Kunde, Benutzer etc.) über Faktoren verhandelt werden?

Relevante Fragen zur Flexibilität (am Beispiel *Auslieferung der Funktionalität*):

- Kann der Faktor beeinflusst oder geändert werden, so dass der Architekturentwurf vereinfacht werden kann?
 - ▶ *Auslieferung der Funktionalität ist verhandelbar. Weniger wichtige Funktionalität kann in späterem Release ausgeliefert werden.*
- Wie kann man ihn beeinflussen?
 - ▶ *Nur nach rechtzeitiger Absprache mit dem Kunden.*
- Zu welchem Grad kann man ihn beeinflussen?
 - ▶ *Der Kunde hat Mindestanforderungen, die erfüllt werden müssen.*

Flexibilität und Priorisierung

SIMPLY EXPLAINED



Globale Analyse: Analysiere Faktoren

2.b Charakterisiere ihre Veränderlichkeit

Veränderlichkeit

Kann sich der Faktor während der Entwicklung ändern?

Globale Analyse: Analysiere Faktoren

2.b Charakterisiere ihre Veränderlichkeit

Veränderlichkeit

Kann sich der Faktor während der Entwicklung ändern?

Relevante Fragen (am Beispiel *Auslieferung der Funktionalität*):

- In welcher Weise kann sich der Faktor ändern?
 - Prioritäten könnten sich ändern.
 - Anforderungen könnten wegfallen, weil nicht mehr relevant.
 - Anforderungen könnten hinzukommen, weil sich Rahmenbedingungen geändert haben.

Globale Analyse: Analysiere Faktoren

2.b Charakterisiere ihre Veränderlichkeit

Veränderlichkeit

Kann sich der Faktor während der Entwicklung ändern?

Relevante Fragen (am Beispiel *Auslieferung der Funktionalität*):

- In welcher Weise kann sich der Faktor ändern?
 - ▶ Prioritäten könnten sich ändern.
 - ▶ Anforderungen könnten wegfallen, weil nicht mehr relevant.
 - ▶ Anforderungen könnten hinzukommen, weil sich Rahmenbedingungen geändert haben.
- Wie wahrscheinlich ist die Änderung während und nach der Entwicklung?
 - ▶ Moderat während der Entwicklung; sehr hoch danach.

Globale Analyse: Analysiere Faktoren

2.b Charakterisiere ihre Veränderlichkeit

Veränderlichkeit

Kann sich der Faktor während der Entwicklung ändern?

Relevante Fragen (am Beispiel *Auslieferung der Funktionalität*):

- In welcher Weise kann sich der Faktor ändern?
 - ▶ Prioritäten könnten sich ändern.
 - ▶ Anforderungen könnten wegfallen, weil nicht mehr relevant.
 - ▶ Anforderungen könnten hinzukommen, weil sich Rahmenbedingungen geändert haben.
- Wie wahrscheinlich ist die Änderung während und nach der Entwicklung?
 - ▶ Moderat während der Entwicklung; sehr hoch danach.
- Wie oft wird er sich ändern?
 - ▶ Alle 3 Monate.

Globale Analyse: Analysiere Faktoren

2.b Charakterisiere ihre Veränderlichkeit

Veränderlichkeit

Kann sich der Faktor während der Entwicklung ändern?

Relevante Fragen (am Beispiel *Auslieferung der Funktionalität*):

- In welcher Weise kann sich der Faktor ändern?
 - ▶ Prioritäten könnten sich ändern.
 - ▶ Anforderungen könnten wegfallen, weil nicht mehr relevant.
 - ▶ Anforderungen könnten hinzukommen, weil sich Rahmenbedingungen geändert haben.
- Wie wahrscheinlich ist die Änderung während und nach der Entwicklung?
 - ▶ Moderat während der Entwicklung; sehr hoch danach.
- Wie oft wird er sich ändern?
 - ▶ Alle 3 Monate.
- Wird der Faktor betroffen von Änderungen anderer Faktoren?
 - ▶ Änderung technologischer Aspekte (Software-/Hardware-Plattform).

Globale Analyse: Analysiere Faktoren

3. Analysiere die Auswirkungen der Änderung von Faktoren auf Architektur

Auswirkungen

Was wird von dem Faktor *wie* beeinflusst?

Globale Analyse: Analysiere Faktoren

3. Analysiere die Auswirkungen der Änderung von Faktoren auf Architektur

Auswirkungen

Was wird von dem Faktor *wie* beeinflusst?

Auswirkungen auf:

- Andere Faktoren
 - ▶ *Z.B. moderater Einfluss auf Einhaltung des Zeitplans und damit darauf, was zu welchem Zeitpunkt ausgeliefert werden kann*
- Systemkomponenten
 - ▶ *Z.B. geplante Komponenten müssen überarbeitet werden; Komponenten können hinzu kommen oder wegfallen.*
- Operationsmodi des Systems
- Andere Entwurfsentscheidungen
- ...

Faktortabelle zu organisatorischen Aspekten

Faktor	Flexibilität und Veränderlichkeit	Auswirkung
O1 : Entwicklungsplan		
O1.1 : Time-To-Market		
Auslieferung Febr. 2013	Keine Flexibilität	Nicht alle Funktionen können implementiert werden
O1.2 : Auslieferung von Produktfunktionen		
priorisierte Funktionen	Funktionen sind verhandelbar	Die Reihenfolge bei der Implementierung der Funktionen kann sich ändern
O2 : Entwicklungsbudget		
O2.1 : Anzahl Entwickler		
6 Entwickler	Keine neuen Entwickler können eingestellt werden. Entwickler können (temporär) ausfallen.	Moderater Einfluss auf Zeitplan; partielle Implementierung droht

Weitere organisatorische Beispielfaktoren I

O1: Management

- Selbst herstellen versus kaufen
- Zeitplan versus Funktionsumfang
- Geschäftsziele

O2: Personal: Fähigkeiten, Interessen, Stärken, Schwächen mit ...

- Anwendungsdomäne
- Softwareentwurf
- speziellen Implementierungstechniken
- speziellen Analysetechniken

Weitere organisatorische Beispielfaktoren II

O3: Prozesse und Werkzeuge für die Entwicklungsschritte, d.h.:

- Entwicklung (IDE), z.B. *Eclipse*
- Konfigurationsmanagement, z.B. *Subversion*
- Systembau (Build) z.B. *Maven*
- Test, z.B. *JUnit*
- Release, z.B. *Installer*

O4: Entwicklungszeitplan

- Time-To-Market
- Auslieferung der Produktfunktionen
- Release-Zeitplan

Weitere organisatorische Beispielfaktoren III

O5: Entwicklungsbudget

- Anzahl Entwickler
- Kosten von Entwicklungswerkzeugen

Faktortabelle zu technischen Aspekten

Faktor	Flexibilität und Veränderlichkeit	Auswirkung
T2 : Software		
T2.1 : Betriebssystem (BS)		
Bibi-Clients sollen auf verschiedenen BS laufen.	Neuere Versionen alle 3 Jahre	Großer Einfluss auf betriebssystemabhängige Komponenten
T2.2 : Benutzerschnittstelle		
GUI soll sich im BS-spezifischen Look&Feel darstellen	Neuere Versionen alle 2 Jahre	Großer Einfluss auf GUI-Komponenten

Technische Beispielfaktoren I

T1: Hardware

- Prozessoren
- Netzwerk
- Hauptspeicher
- Plattspeicher
- domänenspezifische Hardware

T2: Software

- Betriebssystem
- Benutzerschnittstelle
- Software-Komponenten
- Implementierungssprache
- Entwurfsmuster
- Rahmenwerke

T3: Architekturtechnologie

Technische Beispielfaktoren II

- Architekturstile/-muster und -rahmenwerke
- Domänenspezifische Architekturen oder Referenzarchitekturen
- Architekturbeschreibungssprachen
- Software-Produktlinien-Technologien
- Werkzeuge zur Analyse und Validierung von Architekturen

T4: Standards

- Schnittstelle zum Betriebssystem (z.B. Posix)
- Datenbanken (z.B. SQL)
- Datenformate
- Kommunikation (z.B. TCP/IP)
- Kodierrichtlinien

Faktortabelle zu Produktfaktoren

Faktor	Flexibilität und Veränderlichkeit	Auswirkung
P4 : Verlässlichkeit		
P4.3 : Sicherheit		
hohe Anforderungen an Datensicherheit	nicht flexibel	keine große Auswirkung, wenn Datensicherheit weniger wichtig wird (Architektur könnte vereinfacht werden)

Beispielproduktfaktoren I

P1: Produktfunktionen

P2: Benutzerschnittstelle

- Interaktionsmodell
- Funktionen der Benutzerschnittstelle

P3: Performanz

- Ausführungszeiten
- Speicherbedarf
- Dauer des Systemstarts und -endes
- Wiederherstellungszeit nach Fehlern

Beispielproduktfaktoren II

P4: Verlässlichkeit

- Verfügbarkeit
- Zuverlässigkeit
- Sicherheit

P5: Fehlererkennung, -bericht, -behandlung

- Fehlerklassifikation
- Fehlerprotokollierung
- Diagnostik
- Wiederherstellung

Beispielproduktfaktoren III

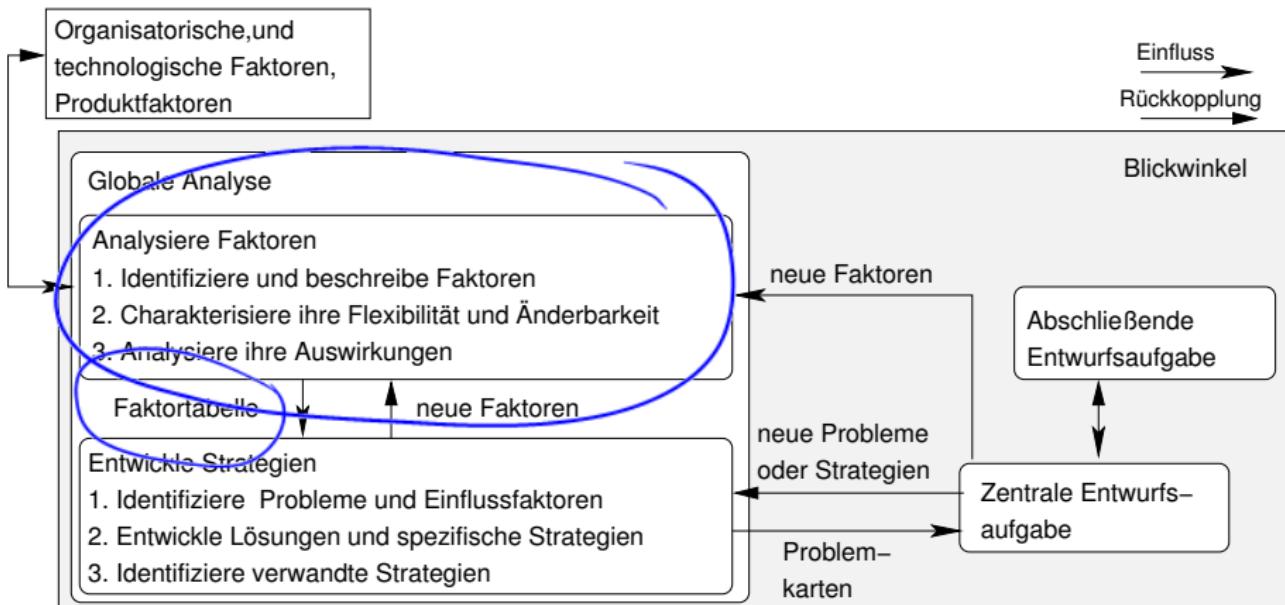
P6: Service

- Service-Dienste (Konfigurations- und Wartungsdienste) für den Betrieb des Systems
- Installation und Aktualisierung
- Test der Software
- Wartung und Erweiterung der Systemimplementierung

P7: Produktkosten

- Hardwarebudget
- Softwarelizenzbudget (für verwendete Software)

Globale Analyse



Globale Analyse: Entwickle Strategien I

1. Identifizierte Probleme und deren Einflussfaktoren

Analysiere Faktorentabelle:

- Grenzen oder Einschränkungen durch Faktoren
 - ▶ *Unverrückbarer Abgabetermin erlaubt keinen vollen Funktionsumfang*
- Notwendigkeit, Auswirkung eines Faktors zu begrenzen
 - ▶ *Entwurf muss Portierbarkeit vorsehen*
- Schwierigkeit, einen Produktfaktor zu erfüllen
 - ▶ *Speicher- und Prozessorbegrenzung erlaubt keine beliebig komplexen Algorithmen*
- Notwendigkeit einer allgemeinen Lösung zu globalen Anforderungen wie Fehlerbehandlung und Wiederaufsetzen nach Fehlern

Globale Analyse: Entwickle Strategien II

2. Entwickle Lösungen und spezifische Strategien

... für die Behandlung der Probleme, die sich implementieren lassen und die notwendige Änderbarkeit unterstützen.

Strategie muss konsistent sein zu:

- Einflussfaktor,
- dessen Veränderlichkeit
- und dessen Interaktion mit anderen Faktoren.

Globale Analyse: Entwickle Strategien III

2. Entwickle Lösungen und spezifische Strategien

Ziele:

- Reduzierung oder Kapselung des Faktoreinflusses
- Reduzierung der Auswirkung einer Änderung des Faktors auf den Entwurf und andere Faktoren
- Reduzierung oder Kapselung notwendiger Bereiche von Expertenwissen oder -fähigkeiten
- Reduzierung der Gesamtentwicklungszeit und -kosten

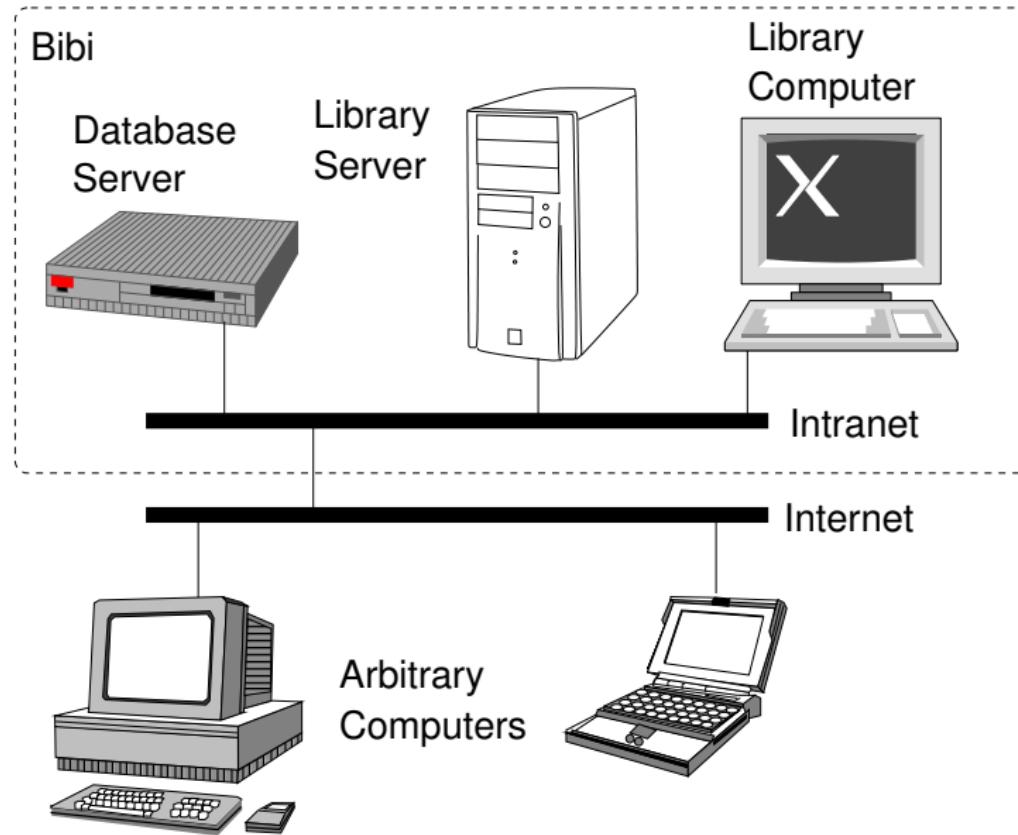
3. Identifizierte verwandte Strategien

Globale Analyse: Entwickle Strategien IV

Problemkarten (Issue Cards) beschreiben Problem und passende Strategien einmalig:

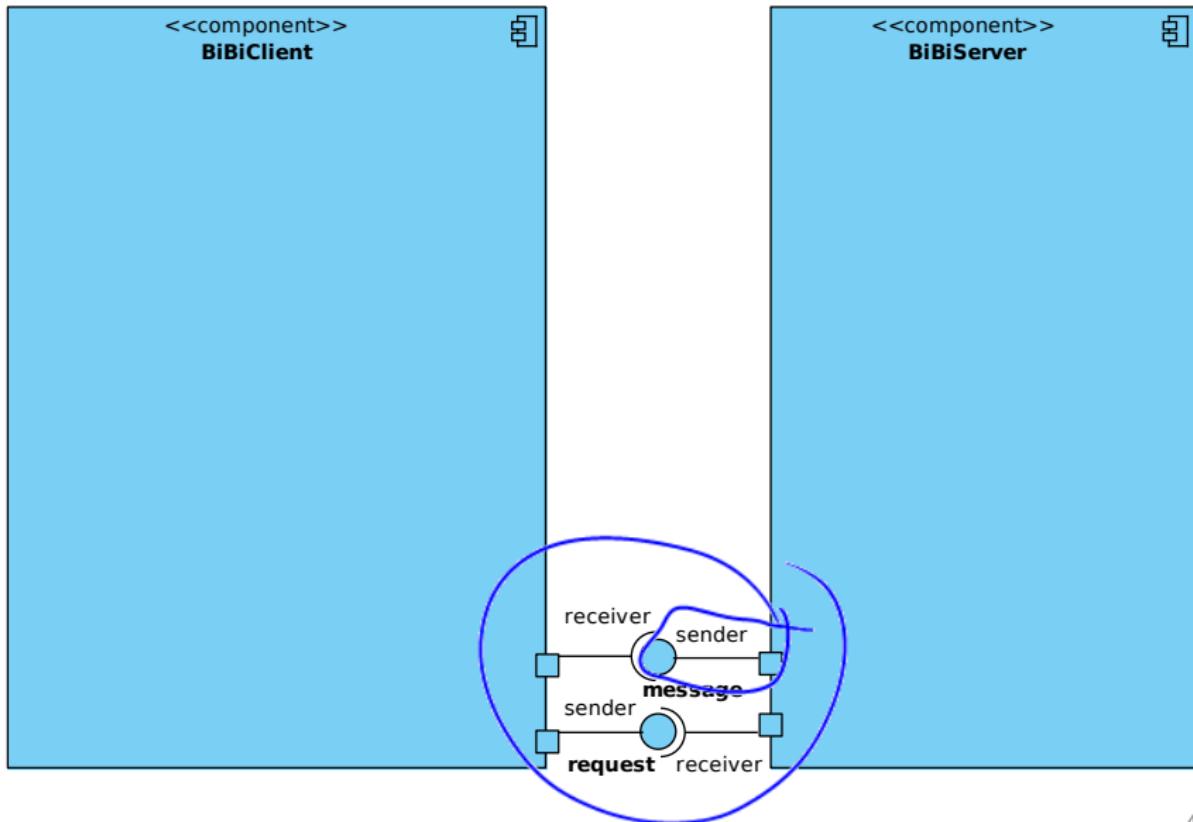
Name des Problems
<i>Beschreibung des Problems</i>
Einflussfaktoren
<i>Liste aller Einflussfaktoren</i>
Lösung
<i>Diskussion einer allgemeinen Lösung</i>
Strategie: Name der Strategie A
<i>Erläuterung der Strategie A</i>
Strategie: Name der Strategie B
<i>Erläuterung der Strategie B</i>
...
Verwandte Strategien
<i>Referenzen zu verwandten Strategien.</i>
<i>Diskussion, in welcher Weise sie verwandt sind.</i>

Beispielsystem Bibi



Strategie

Lege externe Schnittstellen fest.



Netzwerkanpassung

Weiterentwicklung der Netzwerke und Wechsel auf neue Technologie machen Anpassungen an kommunikationsspezifischen Komponenten notwendig. Für die Wartung steht nur ein Entwickler zur Verfügung. Die Anpassungen müssen schnell vorgenommen werden.

Einflussfaktoren

O1.1: Time-To-Market

O2.1: Anzahl Entwickler

T1.2: Betriebssystem

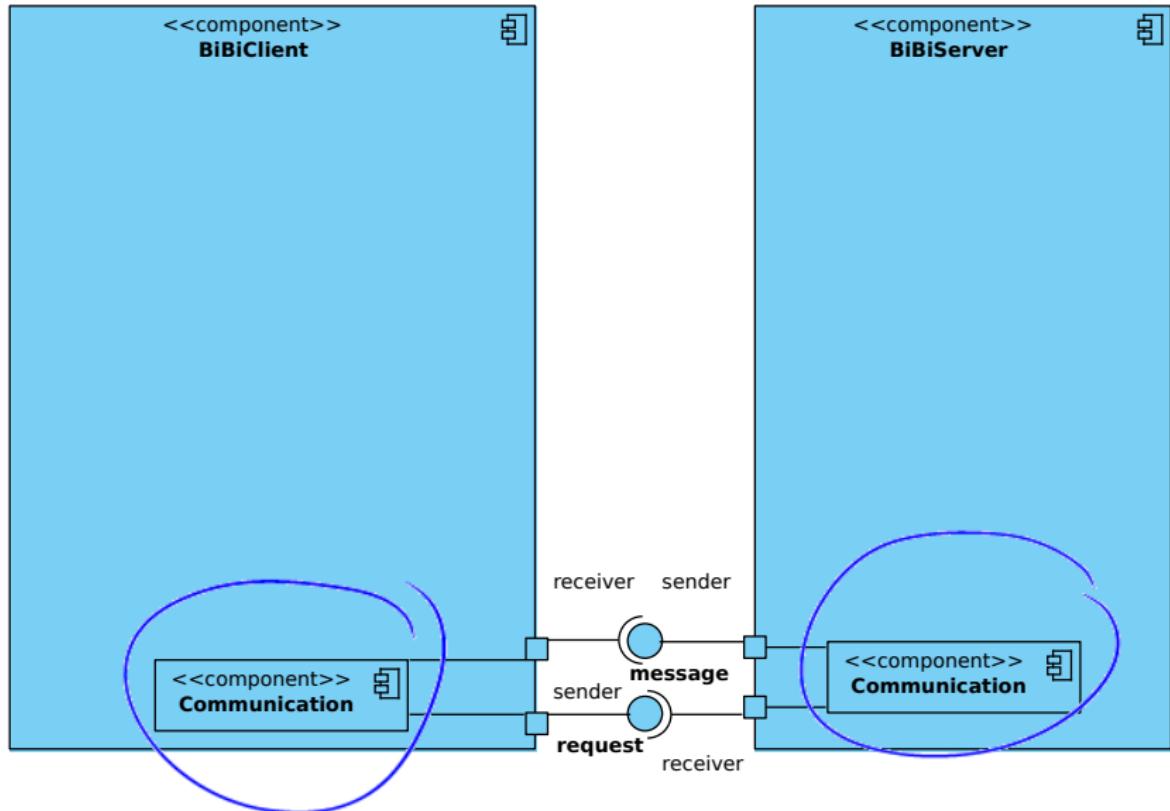
Lösung

Strategie: Kapselung der kommunikationsabhängigen Anteile

Kommunikationsabhängige Anteile werden in Komponenten isoliert. Diese bilden eine virtuelle Maschine für alle anderen Komponenten.

Strategie

Kapselung der kommunikationsabhängigen Anteile



Datensicherheit

Zugriff auf die Daten muss kontrolliert werden. Daten müssen gegen Verlust gesichert werden. Selbst implementierte Lösungen sind riskant (mangelnde Entwicklererfahrung, nicht genug Budget).

Einflussfaktoren

P4.3: Datensicherheit.

O1.1: Time-To-Market

O2.1: Anzahl Entwickler

Lösung

Strategie: Kapselung der Datenhaltung:

Datenhaltung ist eigene Komponente.

Strategie: Wiederverwendung

Einbindung externer wiederverwendbarer Komponenten.

Strategie: Benutzung eines Datenbankmanagementsystems (DBMS):

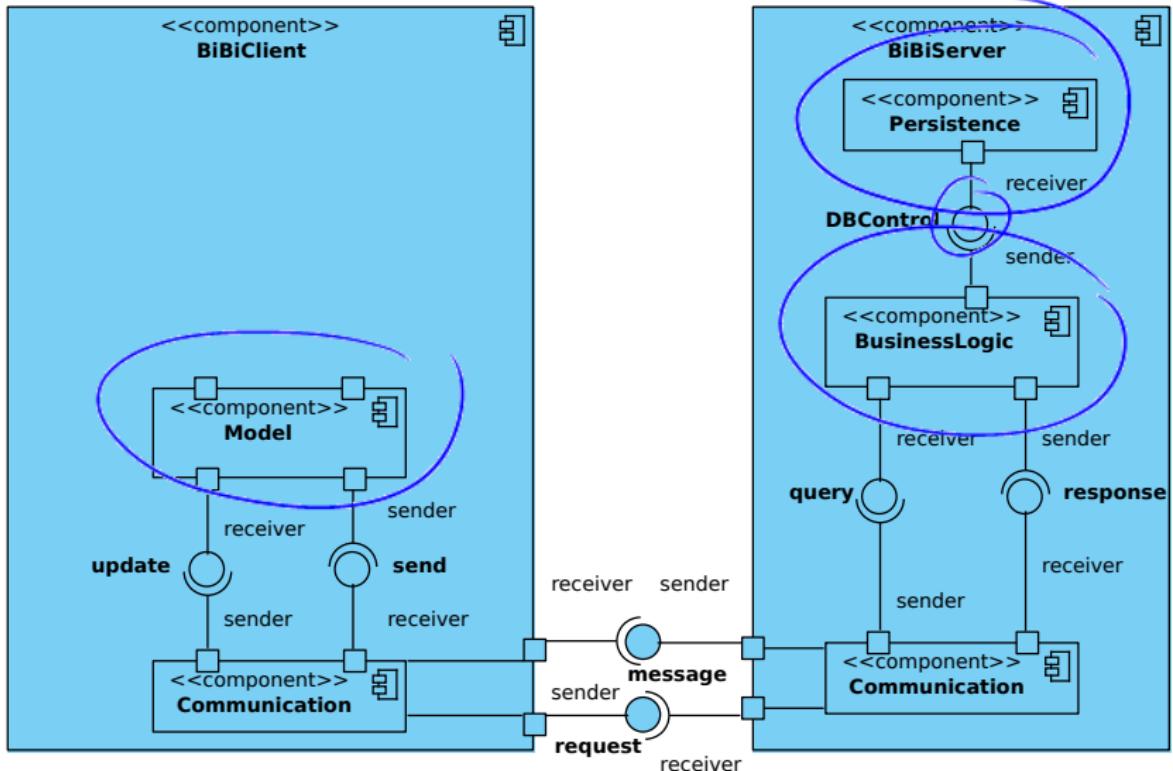
Verwende relationales DBMS.

Strategie: Zugriff auf Daten nur über Server:

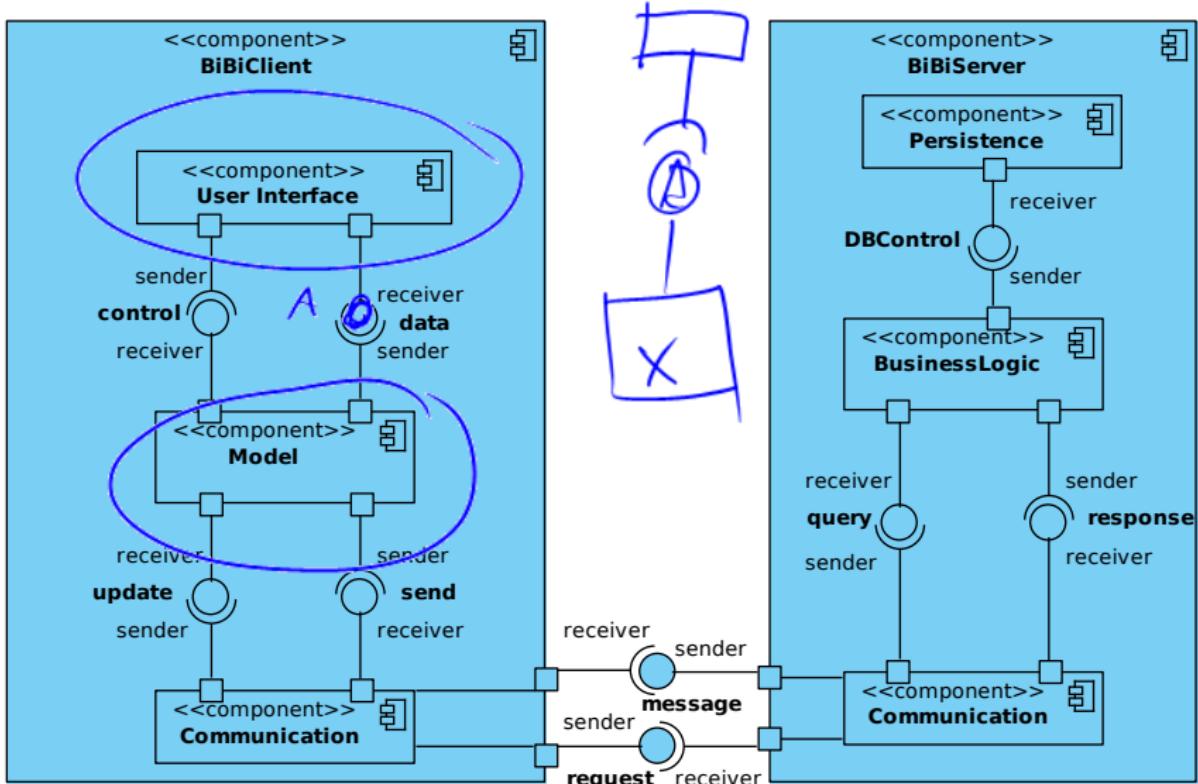
DBMS ist lokal für Server, nicht sichtbar für Clients.

Strategien

Kapselung der Datenhaltung, Wiederverwendung, Benutzung eines DBMS,
Zugriff auf Daten nur über Server



Trennung von Logik und Darstellung



Beispiele

Ambitionierter Zeitplan

Abgabetermin ist fix, Ressourcen sind begrenzt. Möglicherweise können nicht alle Produktfunktionen realisiert werden.

Einflussfaktoren

O1.1 : Time-To-Market

O1.2 : Auslieferung von Produktfunktionen

O2.1 : Anzahl Entwickler

Lösung

Strategie: Schrittweiser Ausbau

System wird schrittweise ausgebaut. Anforderungen werden in der Reihenfolge ihrer Priorität realisiert.

Strategie: Einkaufen statt Selbstentwickeln

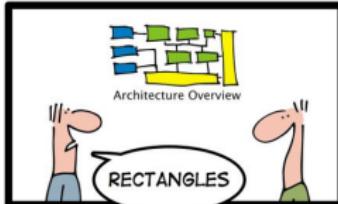
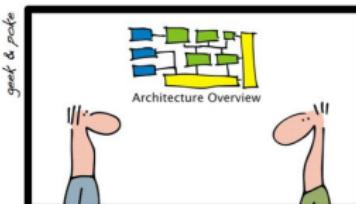
Einbindung externer COTS-Komponenten.

Strategie: Wiederverwendung

Einbindung interner wiederverwendbarer Komponenten.

Architekturdiagramme

ENTERPRISE ARCHITECTURE MADE EASY



PART 1: DON'T MESS WITH THE GORY DETAILS



Fragen



Was sind Architektursichten und welche gibt es?

Architektsichten und -blickwinkel

Definition

Architektsicht (View): Repräsentation eines ganzen Systems aus der Perspektive einer kohärenten Menge von Anliegen (IEEE P1471 2002).

Architektsichten und -blickwinkel

Definition

Architektsicht (View): Repräsentation eines ganzen Systems aus der Perspektive einer kohärenten Menge von Anliegen (IEEE P1471 2002).

Definition

Architekturblickwinkel (Viewpoint): Spezifikation der Regeln und Konventionen, um eine Architektsicht zu konstruieren und zu benutzen (IEEE P1471 2002).

Ein Blickwinkel ist ein Muster oder eine Vorlage, von der aus individuelle Sichten entwickelt werden können, durch Festlegung von

- Zweck,
- adressierte Betrachter
- und Techniken für Erstellung, Gebrauch und Analyse.

Architektsichten und -blickwinkel

Definition

Architektsicht (View): Repräsentation eines ganzen Systems aus der Perspektive einer kohärenten Menge von Anliegen (IEEE P1471 2002).

Definition

Architekturblickwinkel (Viewpoint): Spezifikation der Regeln und Konventionen, um eine Architektsicht zu konstruieren und zu benutzen (IEEE P1471 2002).

Ein Blickwinkel ist ein Muster oder eine Vorlage, von der aus individuelle Sichten entwickelt werden können, durch Festlegung von

- Zweck,
- adressierte Betrachter
- und Techniken für Erstellung, Gebrauch und Analyse.

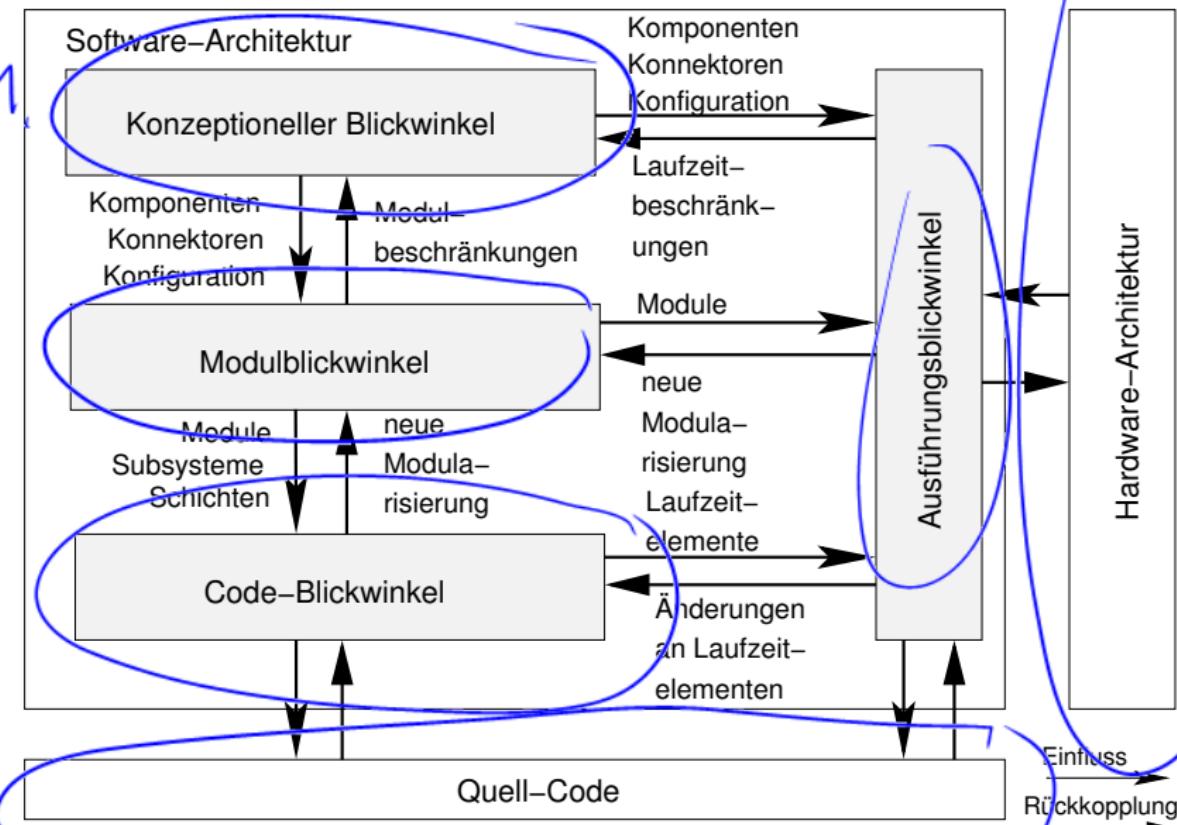
Unterschiedliche Sichten helfen der Strukturierung: Separation of Concerns.

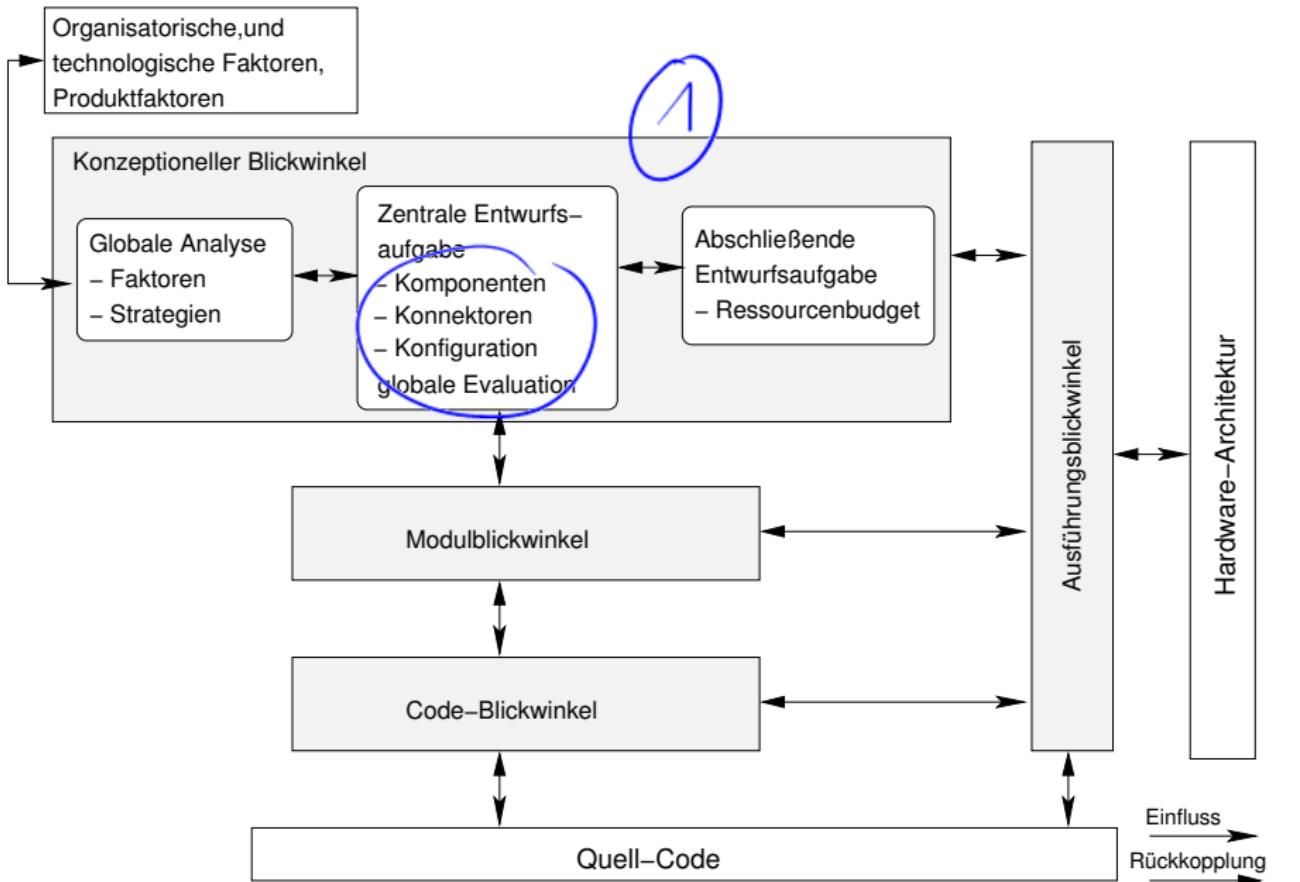
Siemens-Blickwinkel (Hofmeister u. a. 2000)



- **Konzeptioneller Blickwinkel:** beschreibt logische Struktur des Systems; abstrahiert weitgehend von technologischen Details
- **Modulblickwinkel:** beschreibt die statische logische Struktur des Systems
- **Ausführungsblickwinkel:** beschreibt die dynamische logische Struktur des Systems
- **Code-Blickwinkel:** beschreibt die „anfassbaren“ Elemente des Systems (Quelldateien, Bibliotheken, ausführbare Dateien etc.)

Siemens-Blickwinkel (Hofmeister u. a. 2000)





Konzeptioneller Blickwinkel (Hofmeister u. a. 2000)

- ist der Anwendungsdomäne am nächsten
- Systemfunktionalität wird abgebildet auf
 - ▶ **Konzeptionelle Komponenten**: rechenbetonte Elemente oder Datenhaltung
 - ▶ **Konnektoren**: Kontroll- und Datenfluss zwischen konzeptionellen Komponenten

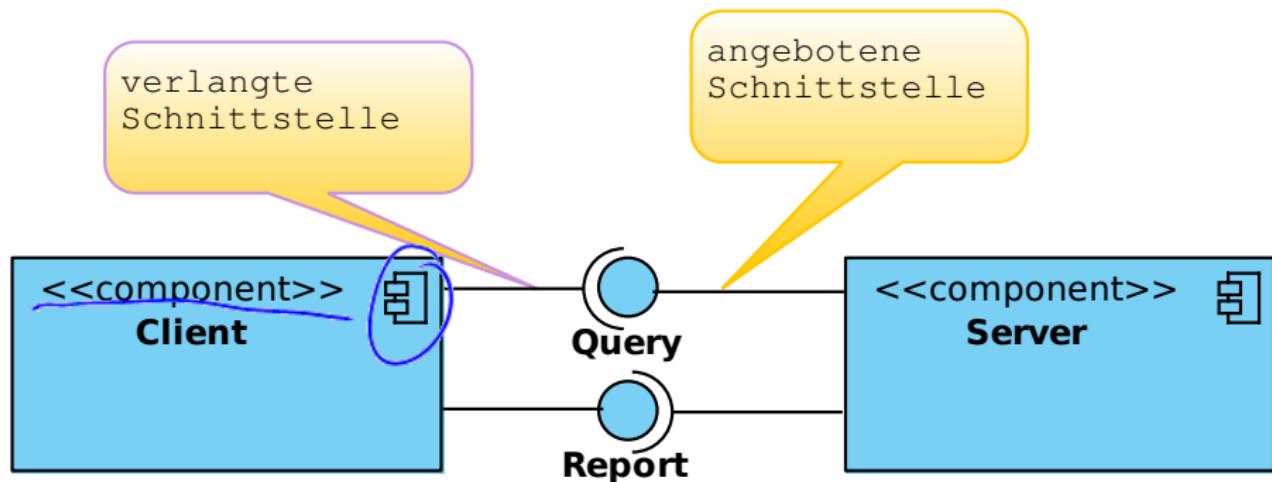
Konzeptioneller Blickwinkel (Hofmeister u. a. 2000)

- ist der Anwendungsdomäne am nächsten
- Systemfunktionalität wird abgebildet auf
 - **Konzeptionelle Komponenten**: rechenbetonte Elemente oder Datenhaltung
 - **Konnektoren**: Kontroll- und Datenfluss zwischen konzeptionellen Komponenten

Engineering-Belange:

- Wie erfüllt das System seine Anforderungen?
- Wie werden Commercial-off-the-Shelf-Components (COTS) in das System integriert? 
- Wie wird domänenspezifische Soft- und Hardware einbezogen?
- Wie kann die Auswirkung von Anforderungen oder der Anwendungsdomäne minimiert werden?

Konzeptionelle Sicht mit UML-Komponentendiagrammen



UML-Komponentendiagramme

Definition

Komponente: (engl. Component) verkapselte, eigenständige, vollständige und somit austauschbare Einheit.^a

^aHier benutzt als logische Entwurfskomponenten.

UML-Komponentendiagramme

Definition

Schnittstellen (engl. Interface): Menge von Operationen mit zugehörigem Protokoll:

- **angebotene Schnittstelle** (engl. provided Interface) einer Komponente K: Schnittstelle, die für andere bereit gestellt wird und von K implementiert wird.
- **verlangte Schnittstelle** (engl. required Interface) einer Komponente K: Schnittstelle, die K für das eigene Funktionieren voraussetzt.

UML-Komponentendiagramme

Definition

Schnittstellen (engl. Interface): Menge von Operationen mit zugehörigem Protokoll:

- **angebotene Schnittstelle** (engl. provided Interface) einer Komponente K: Schnittstelle, die für andere bereit gestellt wird und von K implementiert wird.
- **verlangte Schnittstelle** (engl. required Interface) einer Komponente K: Schnittstelle, die K für das eigene Funktionieren voraussetzt.

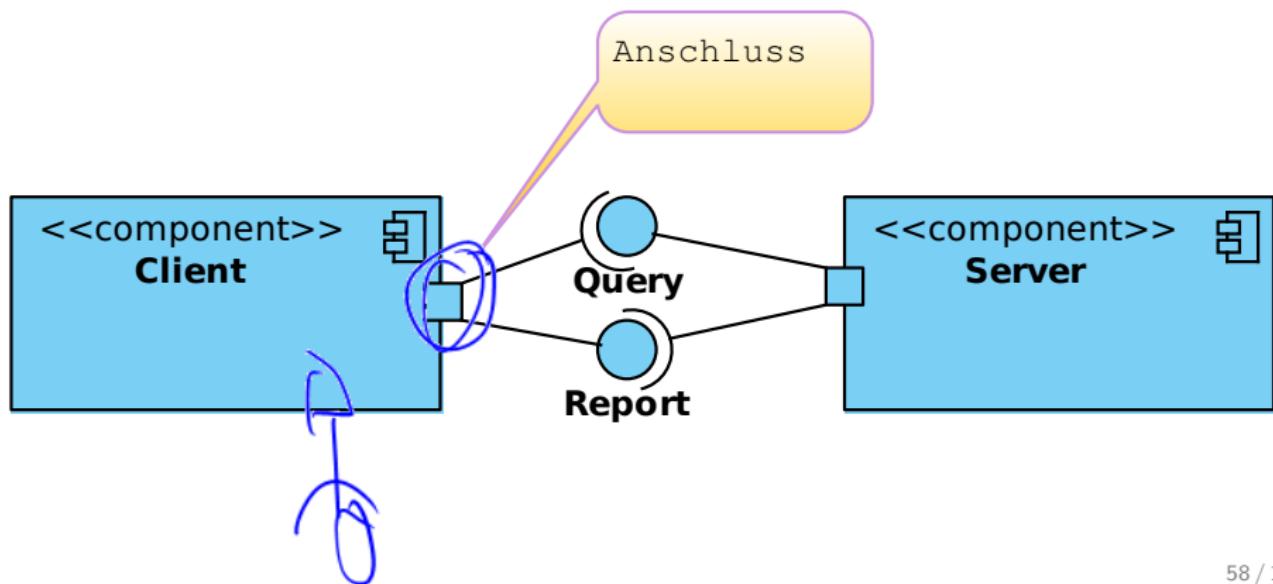
Definition

Protokoll: Spezifikation zulässiger Aufrufe mit Vor- und Nachbedingungen jedes Aufrufs.

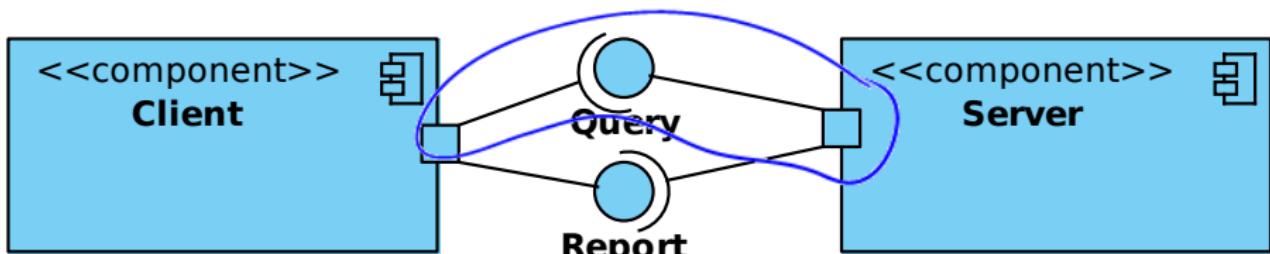
UML-Komponentendiagramme

Definition

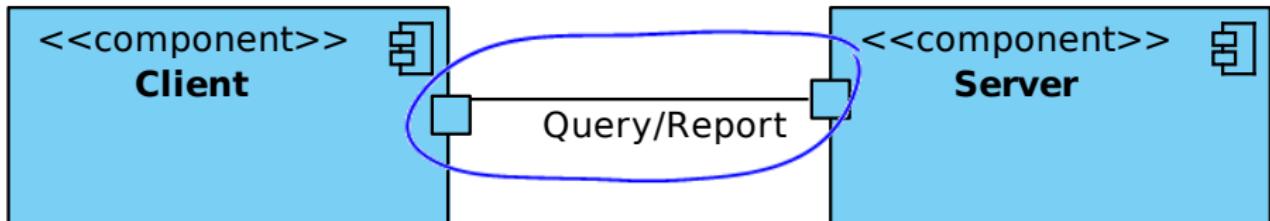
Anschluss stellt Möglichkeit des bidirektionalen Nachrichtenaustausches zwischen zwei zueinander passenden Rollen dar; umfasst angebotene (provided) und verlangte (required) Schnittstellen.



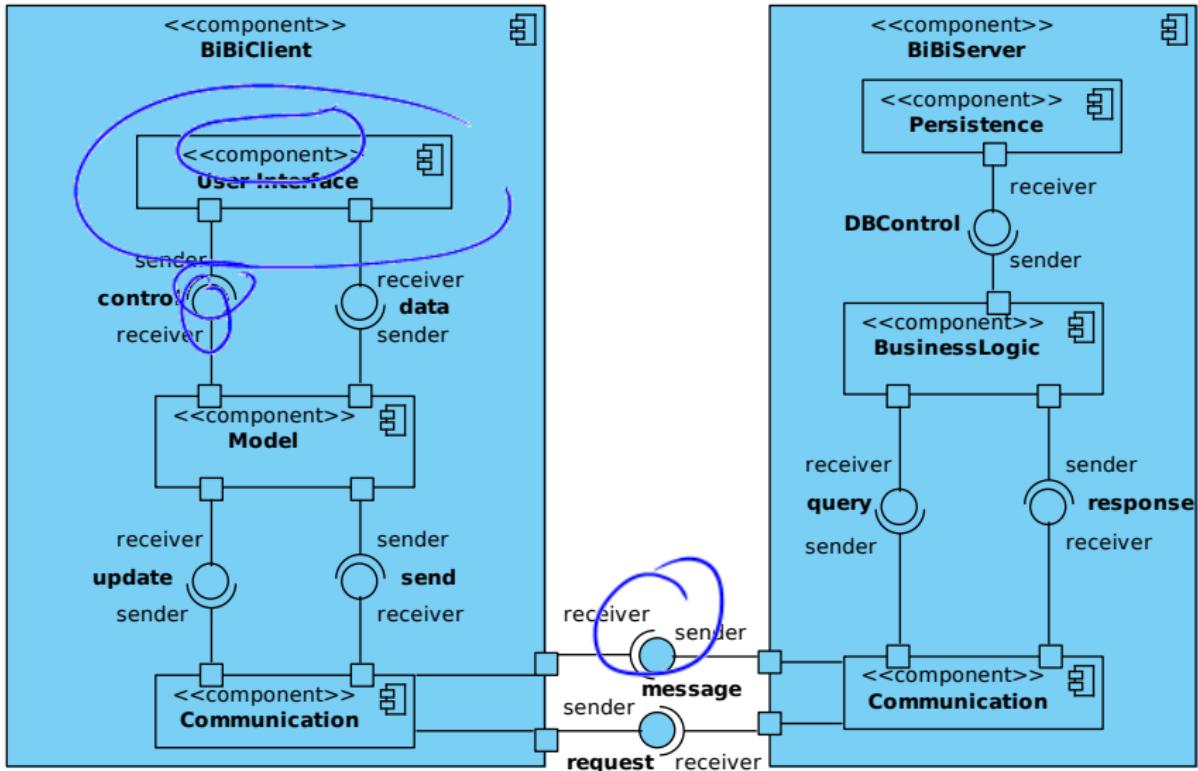
UML-Komponentendiagramme



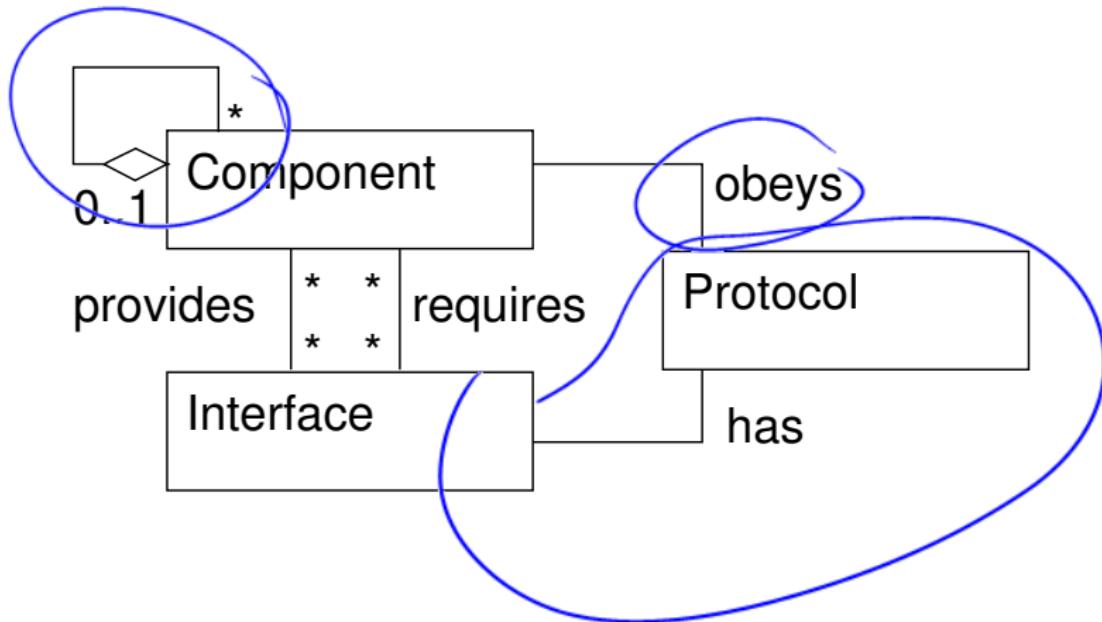
... kann vereinfacht werden zu:



Konzeptionelle Sicht von Bibi



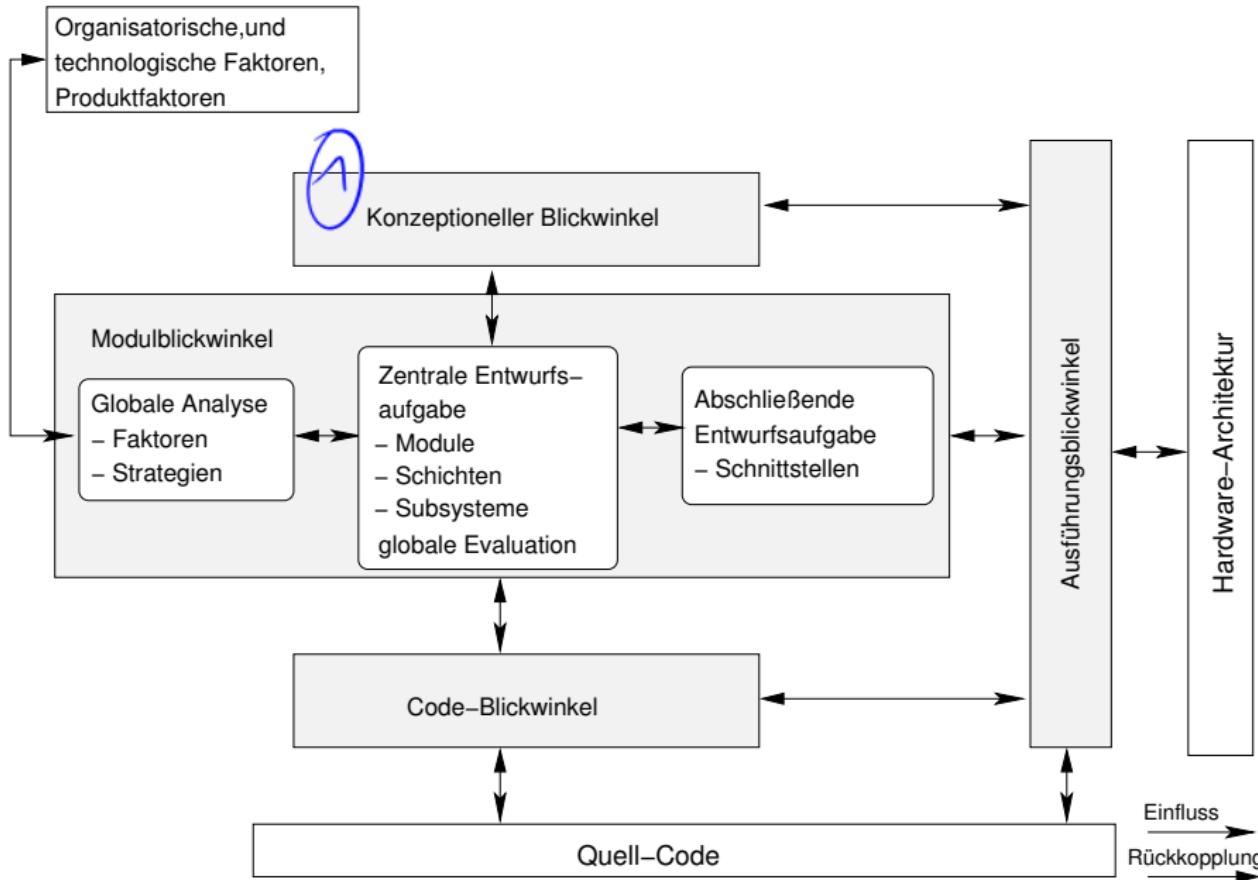
Konzeptioneller Blickwinkel



Fragen



Wie können statische Aspekte (Struktur, Aufbau) einer konzeptionellen Sicht verfeinert werden?



Modulblickwinkel

- bildet Komponenten und Konnektoren auf Module, Schichten und Subsysteme ab
- setzt konzeptionelle Lösung mit aktuellen Softwareplattformen (Programmiersprachen, Bibliotheken) und Technologien um
- beschreibt statische Aspekte

Modulblickwinkel

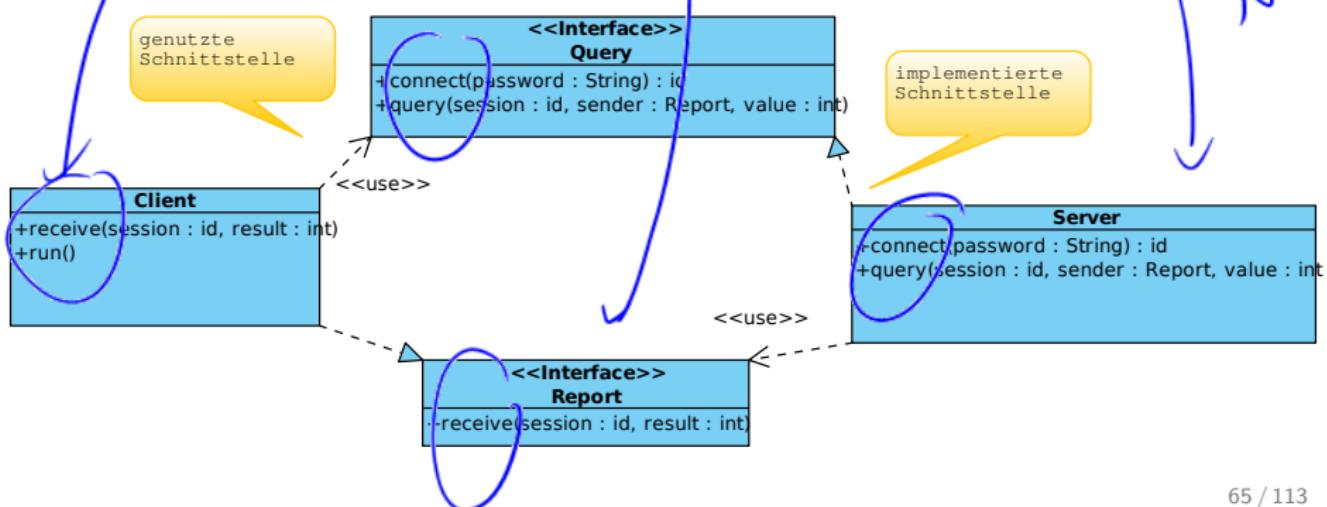
- bildet Komponenten und Konnektoren auf Module, Schichten und Subsysteme ab
- setzt konzeptionelle Lösung mit aktuellen Softwareplattformen (Programmiersprachen, Bibliotheken) und Technologien um
- beschreibt statische Aspekte

Engineering-Belange:

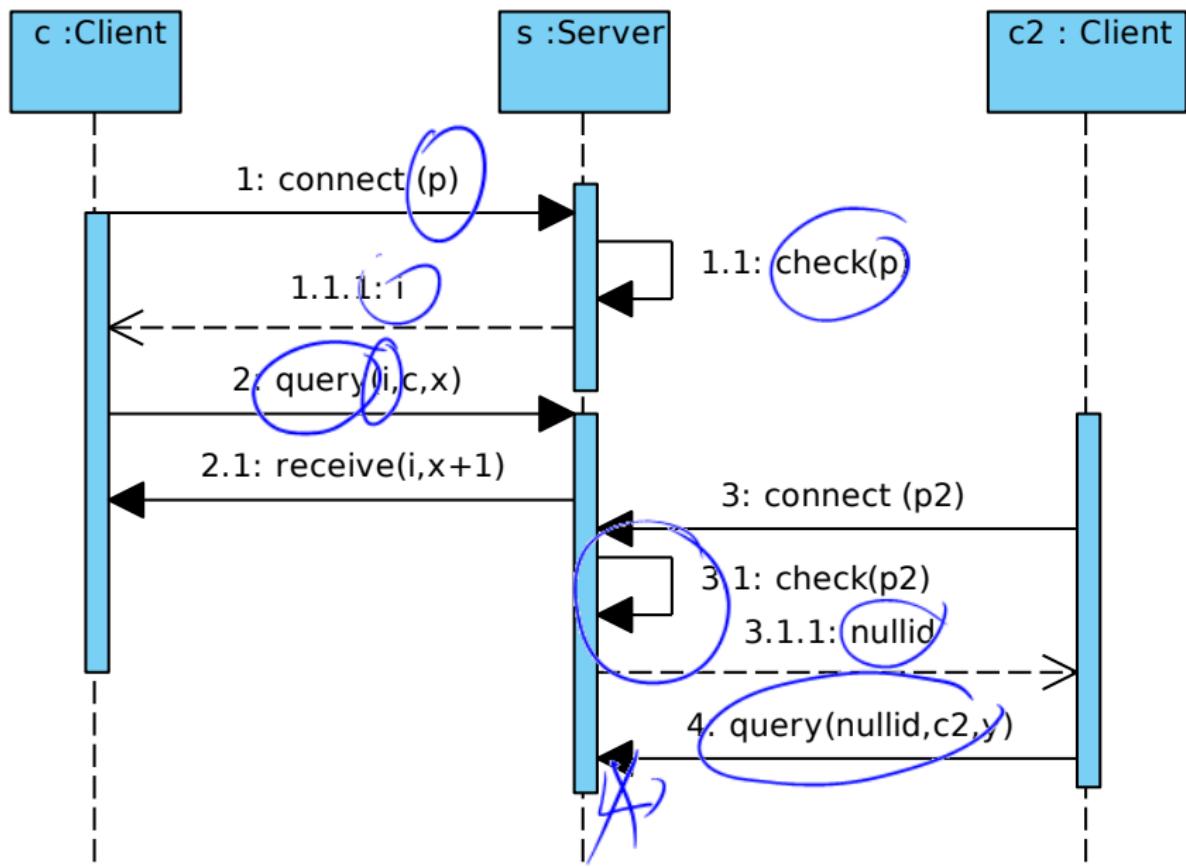
- Wie wird das Produkt auf die Software-Plattform abgebildet?
- Welche Softwaredienste werden benutzt und von wem?
- Wie wird das Testen unterstützt?
- Wie können Abhängigkeiten zwischen Modulen minimiert werden?
- Wie kann die Wiederverwendung von Modulen maximiert werden?
- Welche Techniken können eingesetzt werden, um Auswirkungen von Änderungen zu minimieren?

Vom Abstrakten zum Konkreten

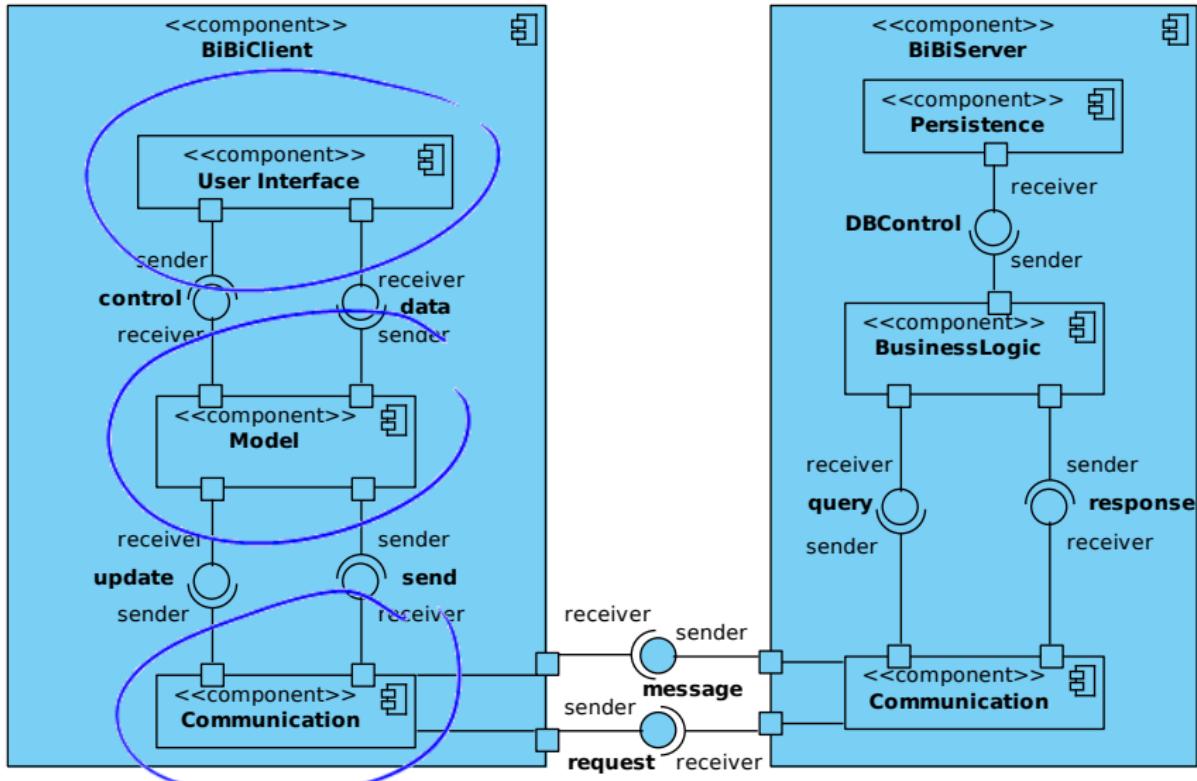
... atomare Komponenten können mit Klassendiagramm konkretisiert werden:



Sequenzdiagramm für das Protokoll

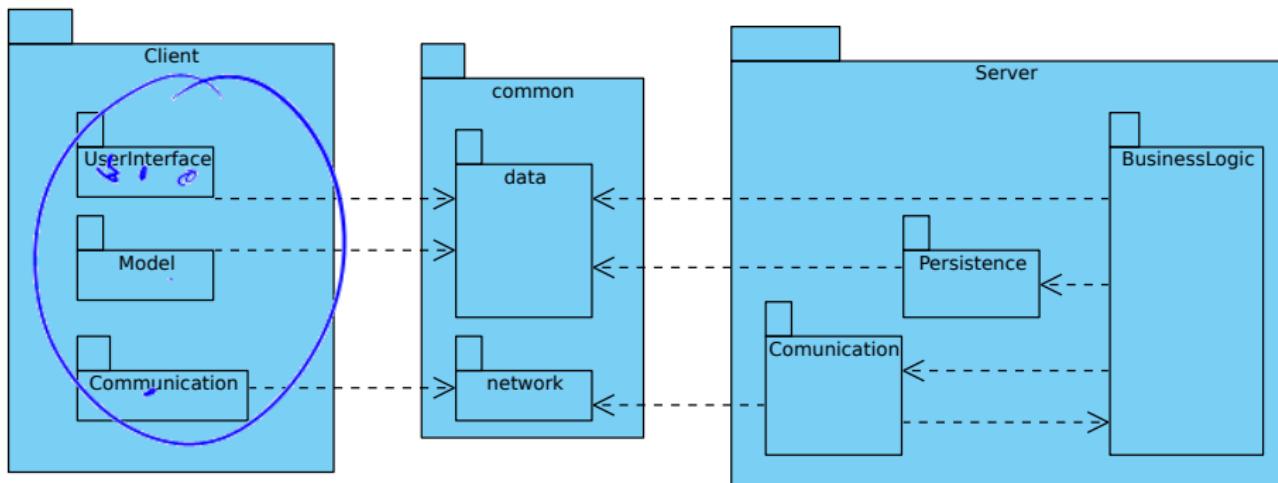


Konzeptionelle Sicht von Bibi



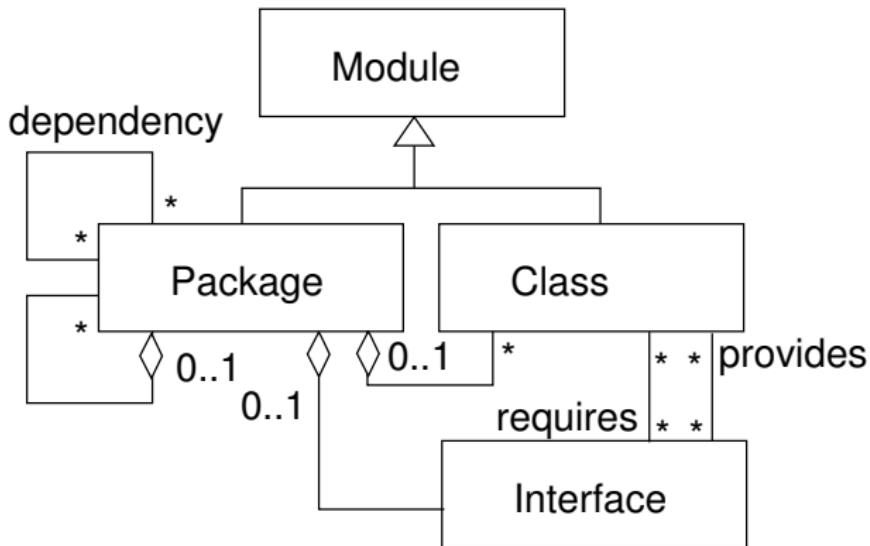
Grobe Modulsicht von Bibi

... zusammengesetzte Komponenten können mit Paketdiagrammen konkretisiert werden (die ihrerseits wieder durch Paket- und Klassendiagramme verfeinert werden):



Modulblickwinkel (Hofmeister u.a. 2000)

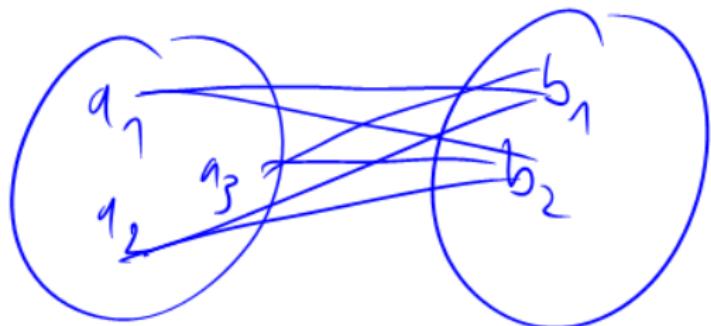
... mit Hilfe von Paket- und Klassendiagrammen

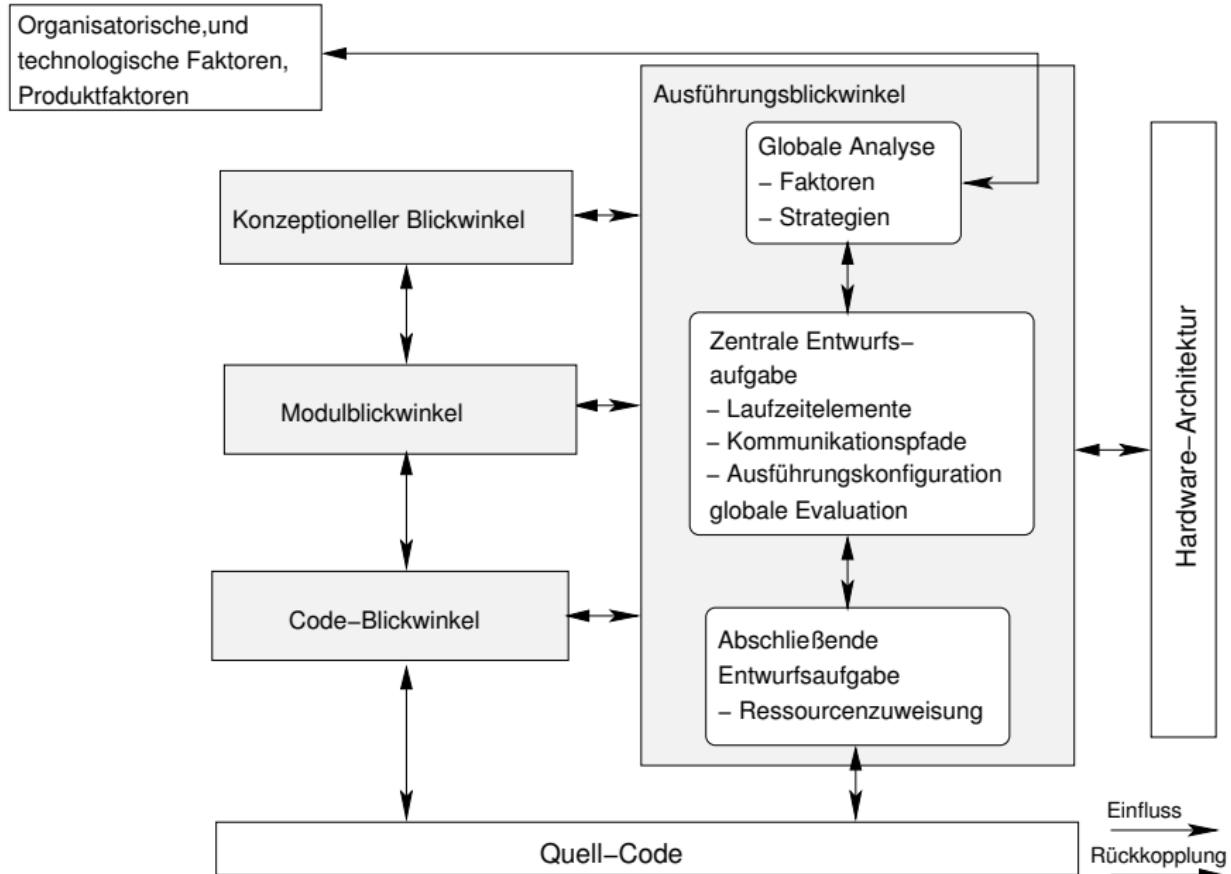


Fragen



Wie kann man die Struktur des Systems zur Laufzeit beschreiben?





Ausführungsblickwinkel

- beschreibt dynamische Aspekte
- beschreibt, wie Module auf Elemente der Ausführungs- und Hardwareplattform abgebildet werden
- definiert Laufzeitelemente und deren Attribute (Speicher- und Zeitbedarf, Allokation auf Hardware)

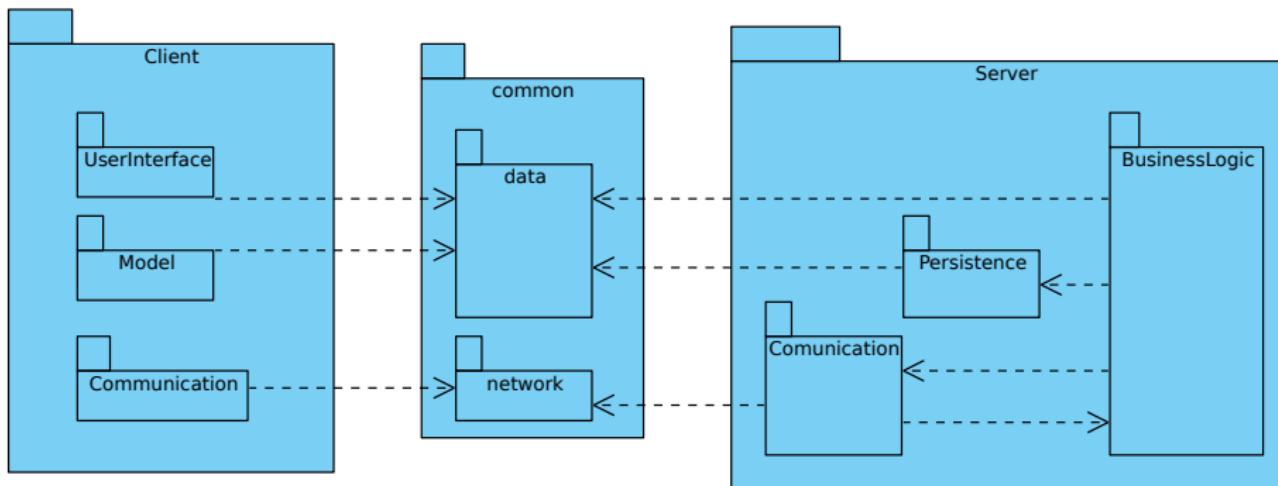
Ausführungsblickwinkel

- beschreibt dynamische Aspekte
- beschreibt, wie Module auf Elemente der Ausführungs- und Hardwareplattform abgebildet werden
- definiert Laufzeitelemente und deren Attribute (Speicher- und Zeitbedarf, Allokation auf Hardware)

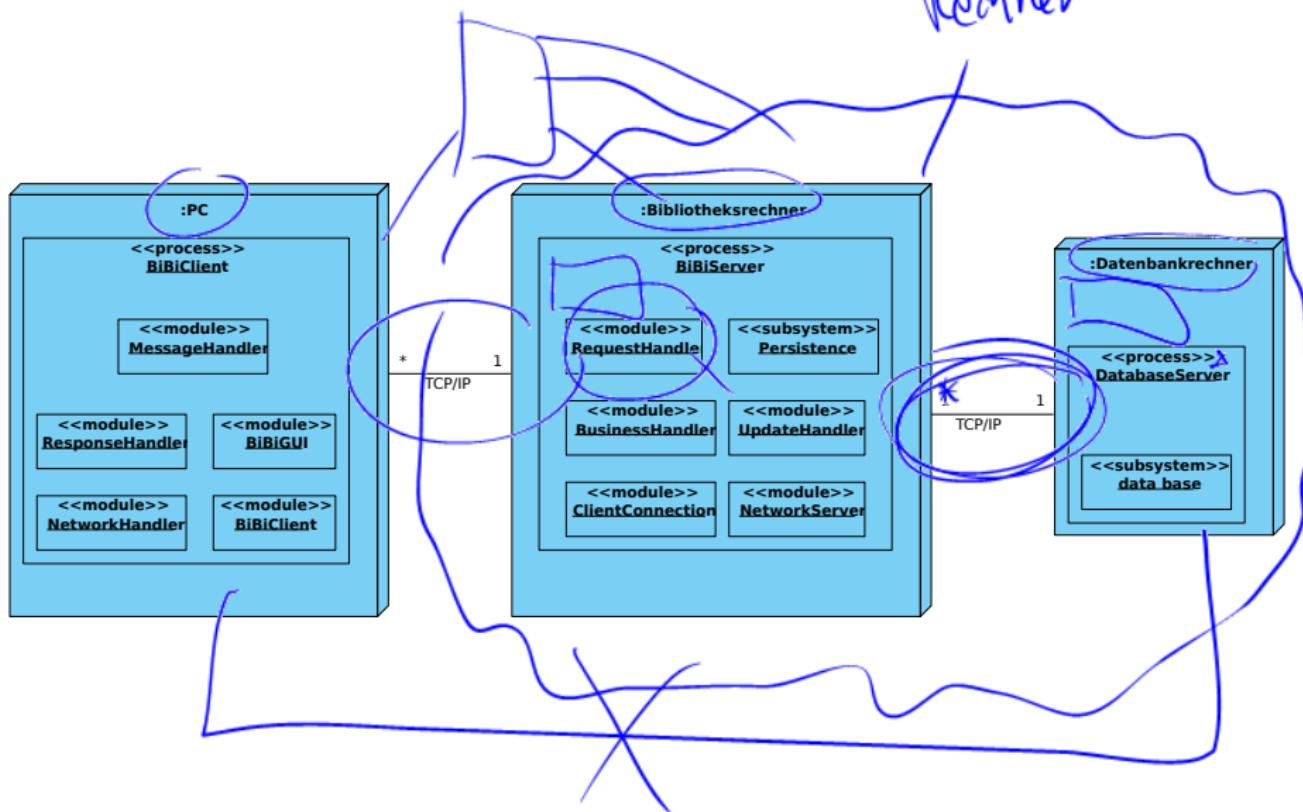
Engineering-Belange:

- Wie verläuft Kontroll- und Datenfluss zwischen Laufzeitkomponenten?
- Wie kann Ressourcenverbrauch ausgewogen werden?
- Wie können Nebenläufigkeit, Replikation und Verteilung erreicht werden, ohne die Algorithmen unnötig zu verkomplizieren?
- Wie können Auswirkungen von Änderungen in der Ausführungsplattform minimiert werden?

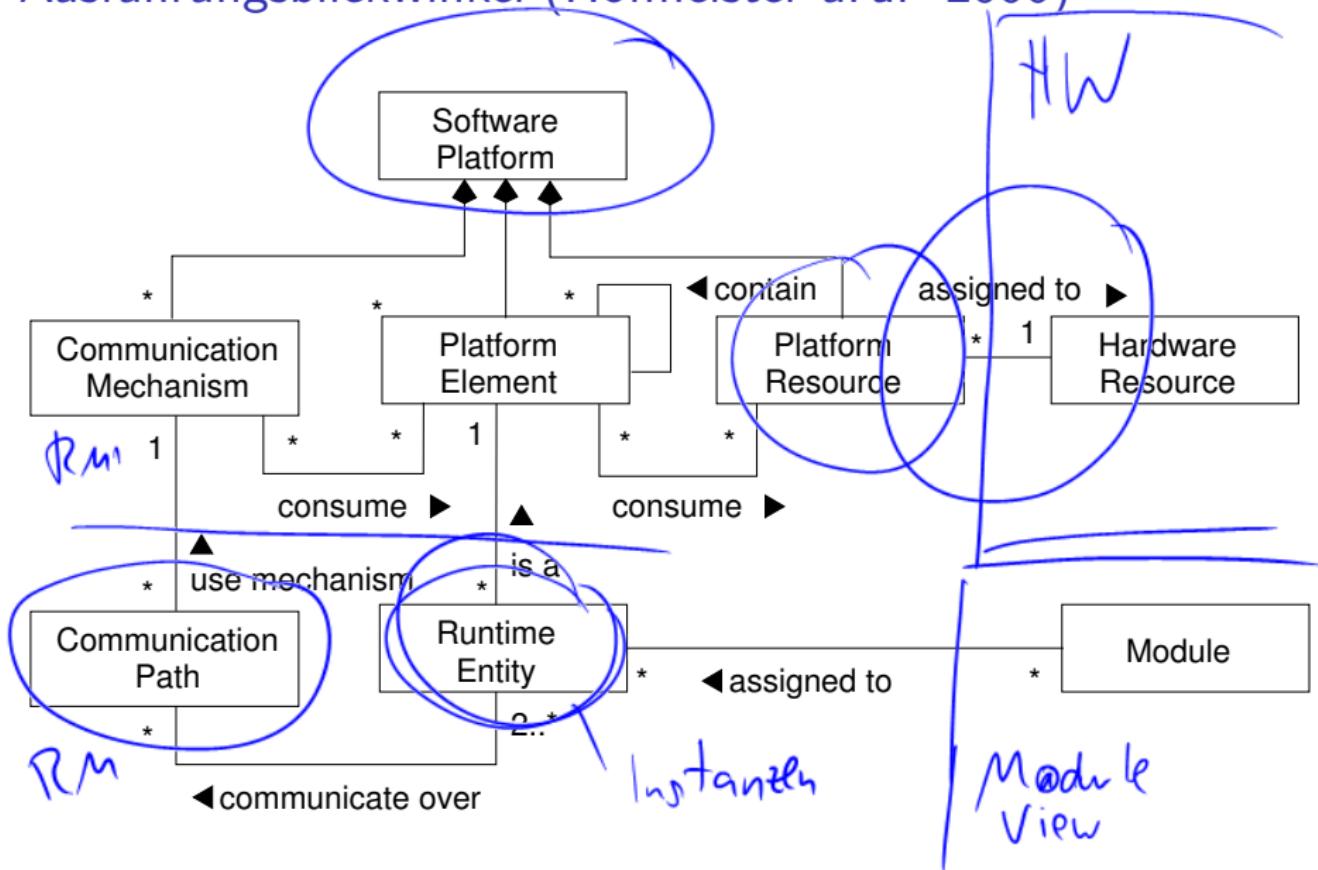
Grobe Modulsicht von Bibi



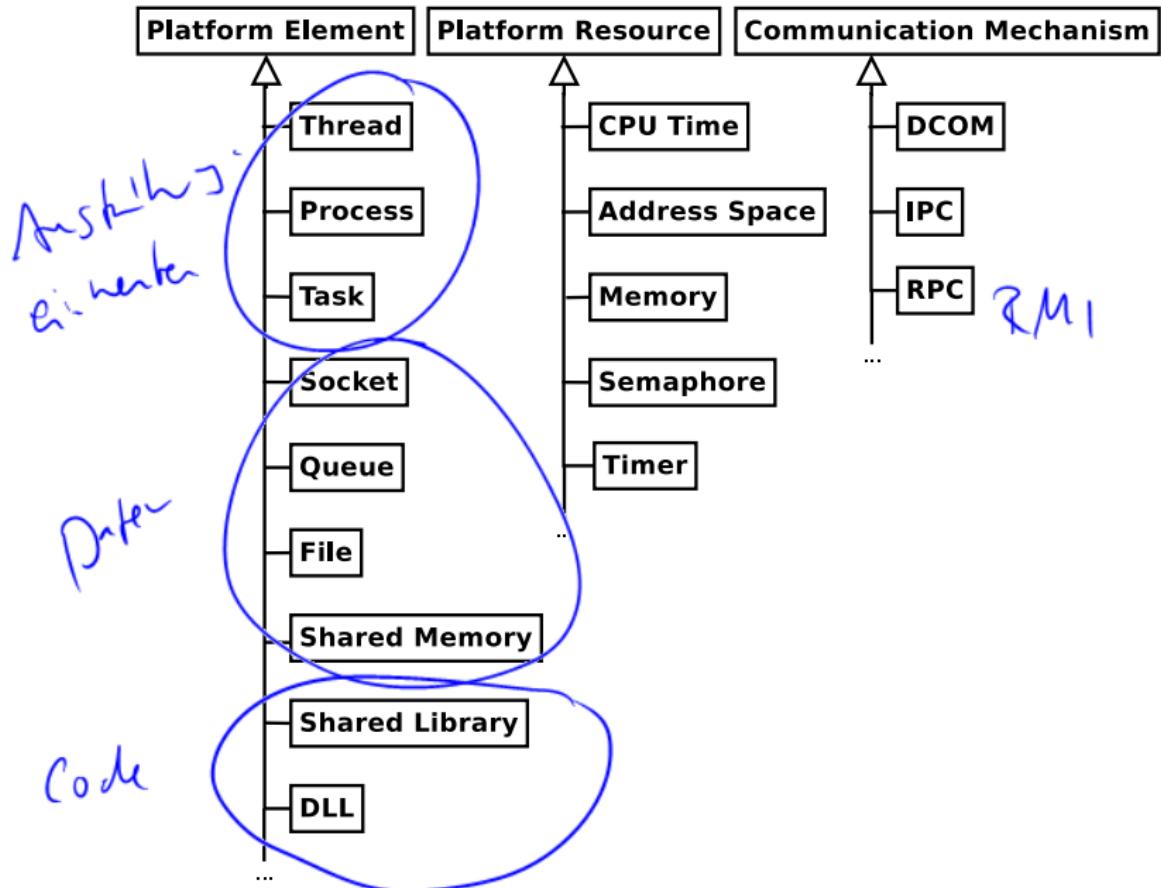
Ausführungsicht von Bibi



Ausführungsblickwinkel (Hofmeister u. a. 2000)



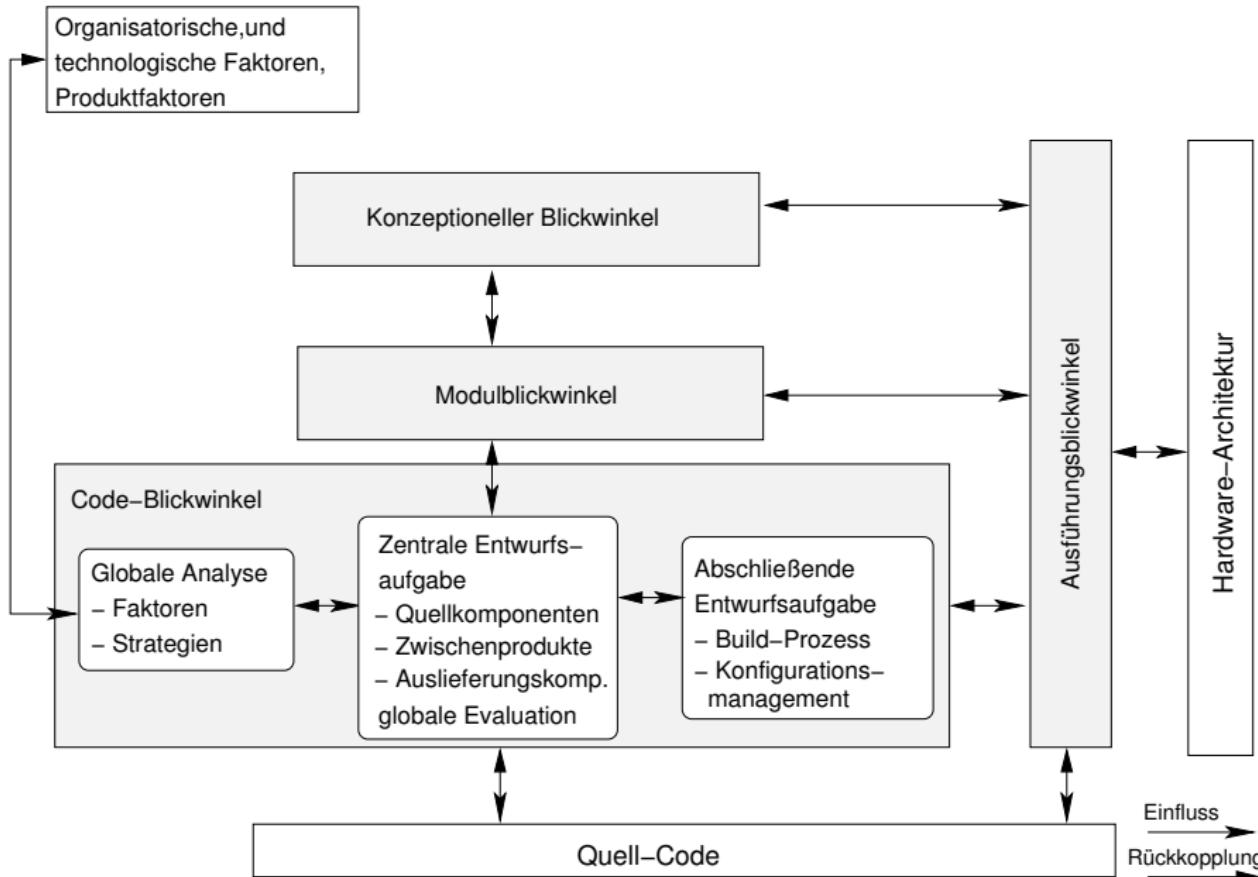
Ausführungsblickwinkel (Hofmeister u. a. 2000)



Fragen



Wie verbindet man Architektur und Code?



Code-Blickwinkel

- bildet Laufzeiteinheiten auf installierbare Objekte (ausführbare Dateien, dynamische Link-Bibliotheken etc.) ab
- bildet Module auf Quellkomponenten (Dateien) ab
- zeigt, wie die installierbaren Dateien aus den Quellkomponenten generiert werden

Code-Blickwinkel

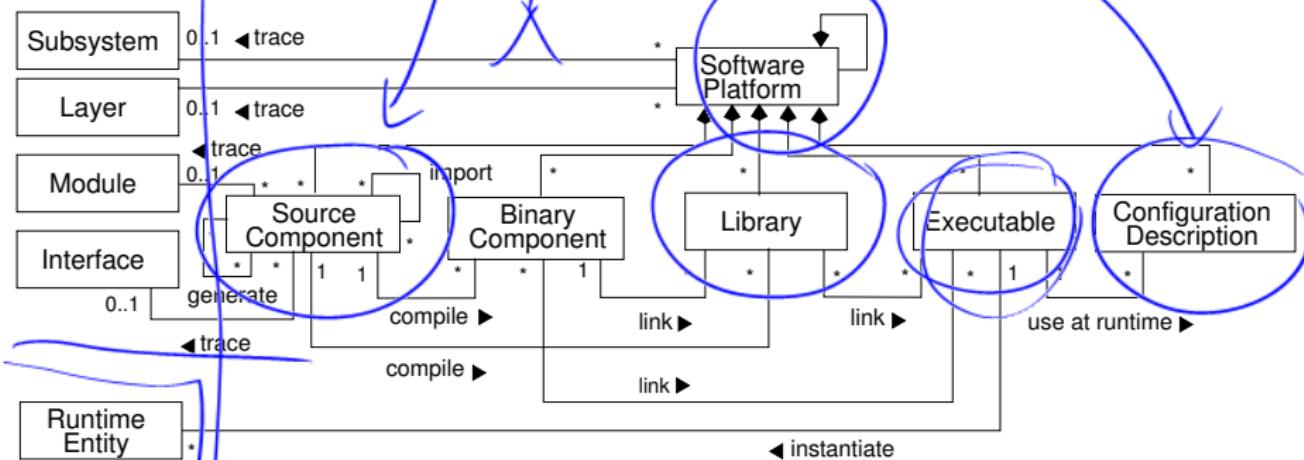
- bildet Laufzeiteinheiten auf installierbare Objekte (ausführbare Dateien, dynamische Link-Bibliotheken etc.) ab
- bildet Module auf Quellkomponenten (Dateien) ab
- zeigt, wie die installierbaren Dateien aus den Quellkomponenten generiert werden

Engineering-Belange:

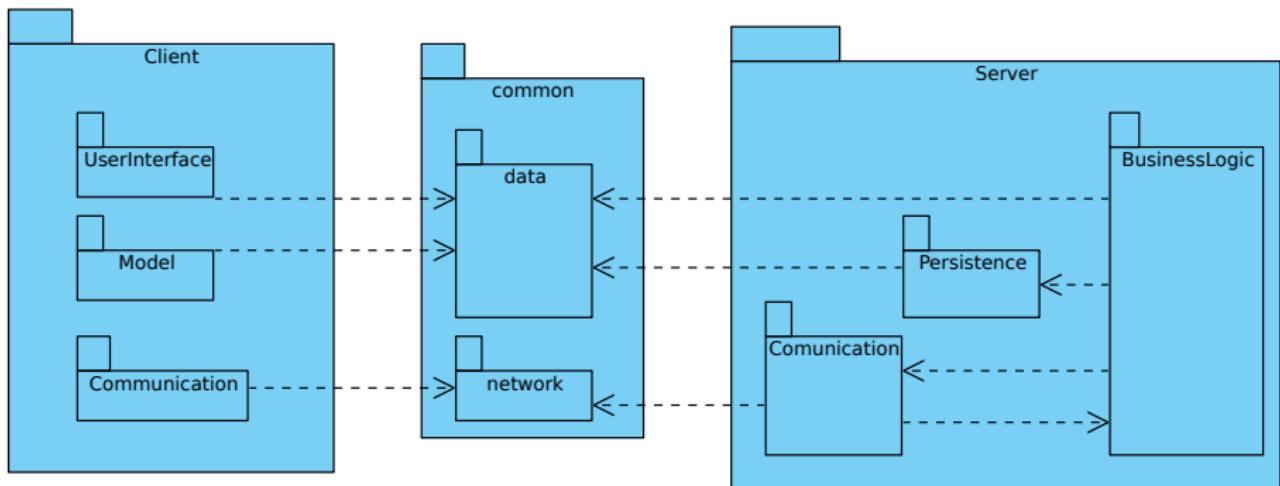
- Wie kann die Zeit und der Aufwand für Produkt-Upgrades verringert werden?
- Wie sollen Produktversionen verwaltet werden?
- Wie kann die Build-Zeit verringert werden?
- Welche Werkzeuge werden in der Entwicklungsumgebung benötigt?
- Wie wird Integration und Test unterstützt?

Code-Blickwinkel (Hofmeister u. a. 2000)

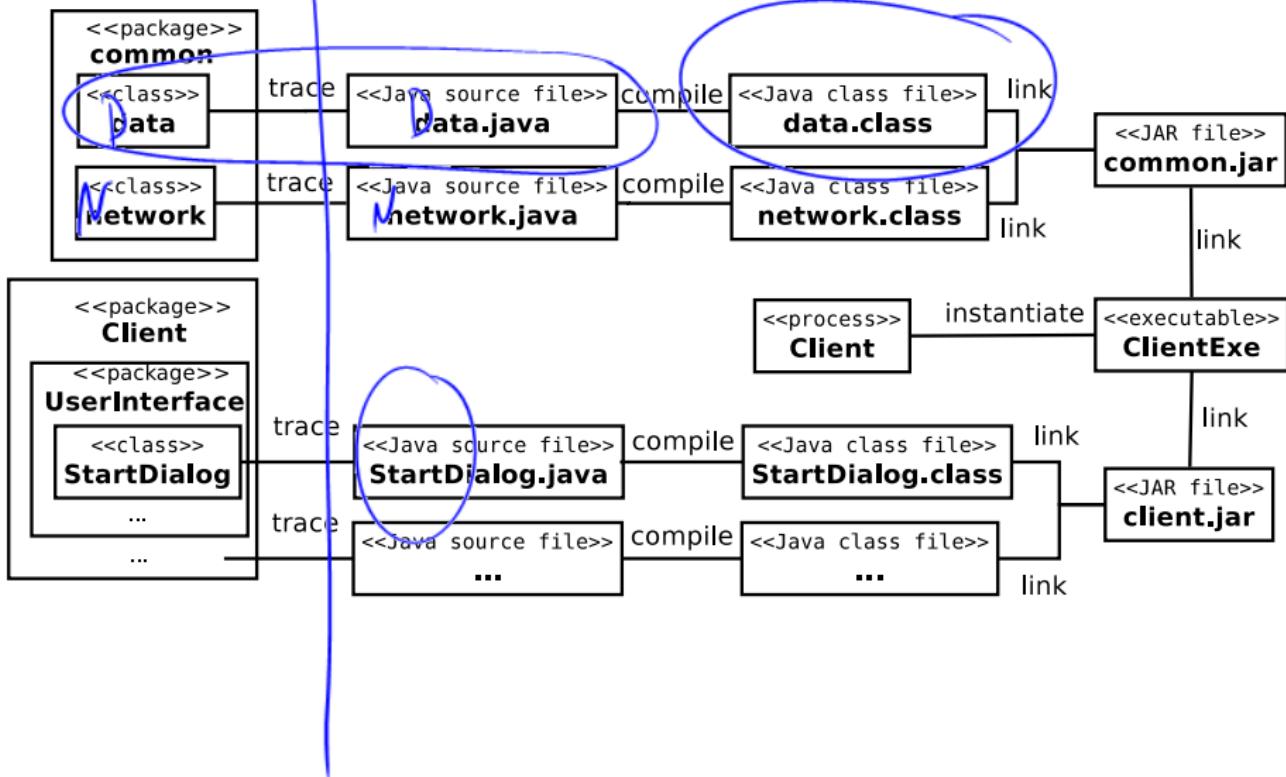
Module



Grobe Modulsicht von Bibi



Code-Blickwinkel: Beispiel



Code-Blickwinkel: Beispiel

Tabellendarstellung:

Module	Source File	JAR
data	data.java	common.jar
network	network.java	common.jar
StartDialog	StartDialog.java	client.jar
...

JAR	instantiates
client.jar	Client
...	...

Fragen



Was sind die angestrebten Qualitäten einer Architektur?

Was ist Modularisierung?

Definition

Modularisierung: Dekomposition eines Systems in Module.

Modul: austauschbares Programmteil, das eine geschlossene Funktionseinheit bildet.

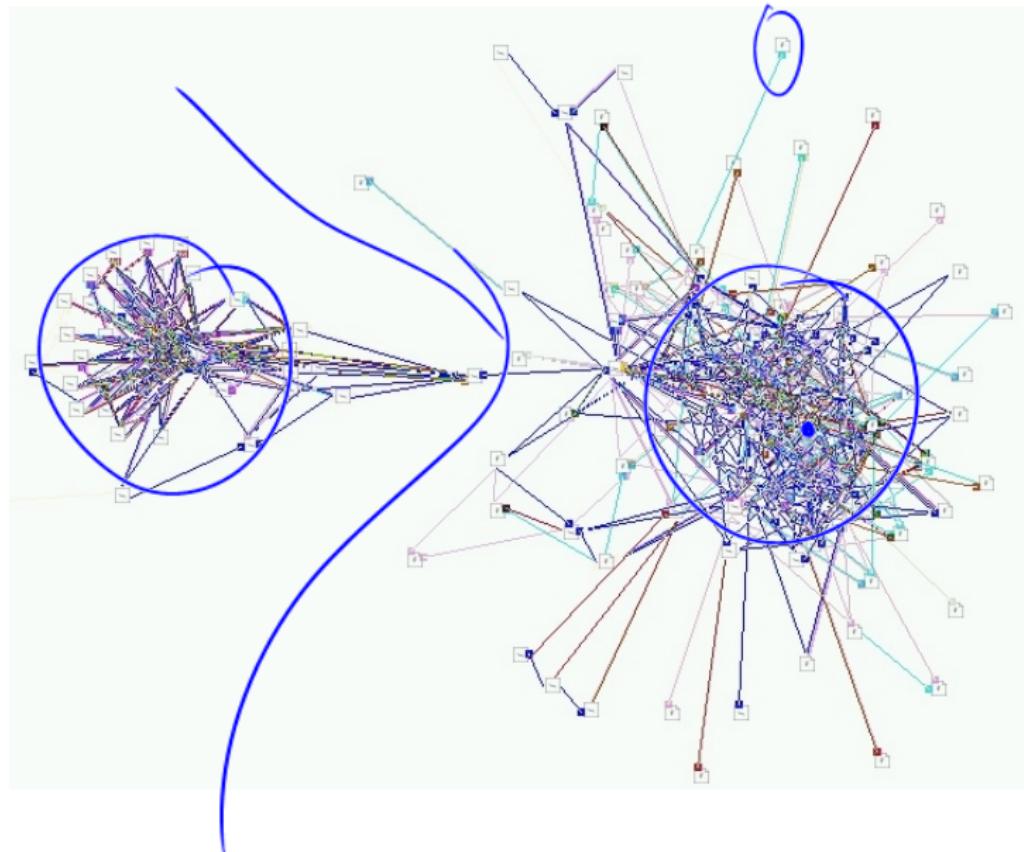
Arbeitspaket: von einer Person oder einer kleinen Gruppe von Entwicklern entwerfbar, implementierbar, testbar, verstehbar, änderbar, ...



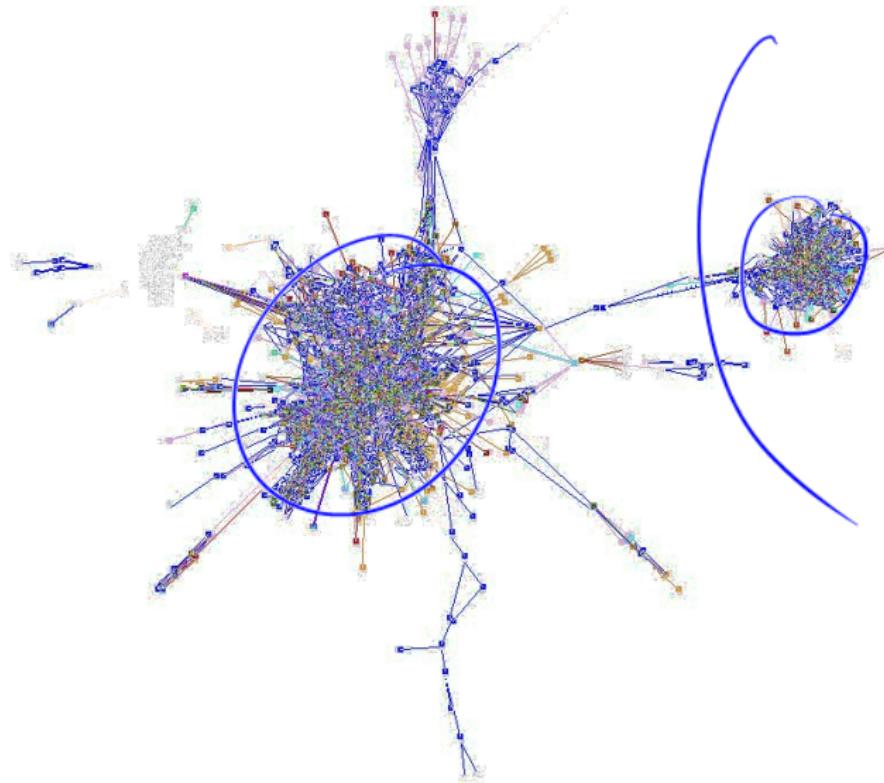
Schlechte Modularisierung



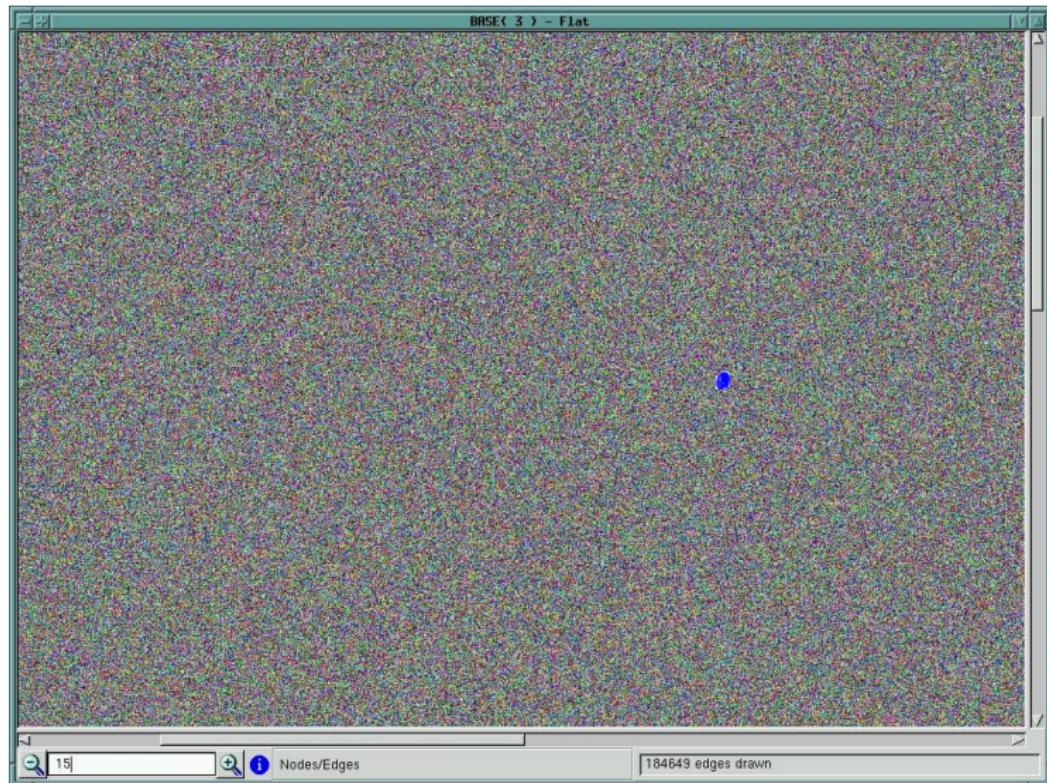
Modularisierung in der Praxis



Modularisierung in der Praxis



Modularisierung in der Praxis



Conways Gesetz:

Conways Gesetz:

Die Struktur der Software spiegelt die Struktur der Organisation wider.

Kopplung und Zusammenhalt

Definition

Kopplung: Grad der Abhangigkeit zwischen Modulen.

Kopplung und Zusammenhalt

Definition

Kopplung: Grad der Abhangigkeit zwischen Modulen.

Definition

Zusammenhalt (Koharenz): Verwandtschaft der Teile eines Moduls.

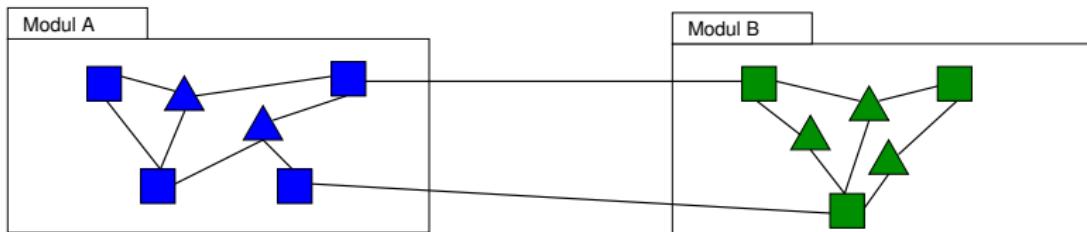
Kopplung und Zusammenhalt

Definition

Kopplung: Grad der Abhangigkeit zwischen Modulen.

Definition

Zusammenhalt (Koharenz): Verwandtschaft der Teile eines Moduls.



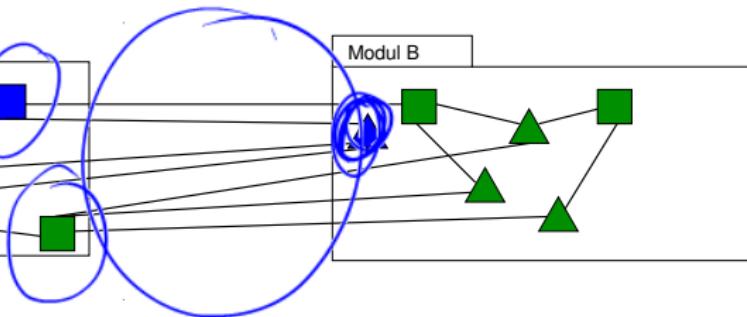
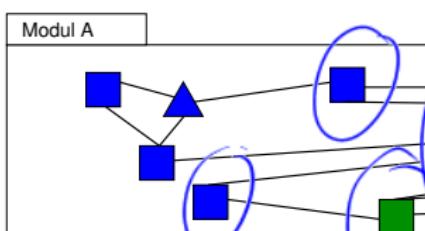
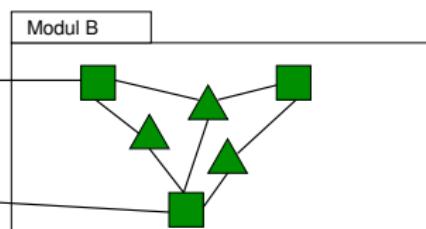
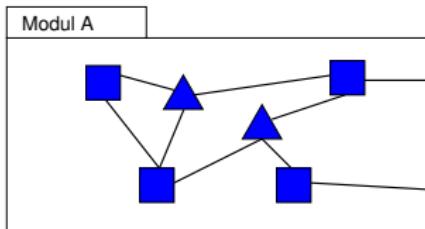
Kopplung und Zusammenhalt

Definition

Kopplung: Grad der Abhangigkeit zwischen Modulen.

Definition

Zusammenhalt (Koharenz): Verwandtschaft der Teile eines Moduls.



Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Moduls	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Moduls	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Moduls	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module
zeitliche Nähe	Verwendung zum selben Zeitpunkt, z.B. Initialisierung, Abschluss	Module, Funktionen

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Moduls	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module
zeitliche Nähe	Verwendung zum selben Zeitpunkt, z.B. Initialisierung, Abschluss	Module, Funktionen
gemeinsame Daten	Zugriff auf bestimmte Daten, exklusiv, z.B. Kalenderpaket (Systemuhr)	Funktionen, Module

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Moduls	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module
zeitliche Nähe	Verwendung zum selben Zeitpunkt, z.B. Initialisierung, Abschluss	Module, Funktionen
gemeinsame Daten	Zugriff auf bestimmte Daten, exklusiv, z.B. Kalenderpaket (Systemuhr)	Funktionen, Module
Hersteller / Verbraucher	ein Teil erzeugt, was der andere verwendet	Module, Funktionen

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Modul	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module
zeitliche Nähe	Verwendung zum selben Zeitpunkt, z.B. Initialisierung, Abschluss	Module, Funktionen
gemeinsame Daten	Zugriff auf bestimmte Daten, exklusiv, z.B. Kalenderpaket (Systemuhr)	Funktionen, Module
Hersteller / Verbraucher	ein Teil erzeugt, was der andere verwendet	Module, Funktionen
einziges Datum	Kapselung einer Datenstruktur, Zusammenfassung der Operationen	Module, Funktionen

Stufen des Zusammenhalts (von schlecht nach gut)

Stufe	innerhalb einer Funktion/Modul	betrifft
kein Zusammenhalt	rein zufällige Zusammenstellung	Module
Ähnlichkeit	z.B. ähnlicher Zweck, also etwa alle Matrixoperationen; auch Fehlerbehandlung	Module
zeitliche Nähe	Verwendung zum selben Zeitpunkt, z.B. Initialisierung, Abschluss	Module, Funktionen
gemeinsame Daten	Zugriff auf bestimmte Daten, exklusiv, z.B. Kalenderpaket (Systemuhr)	Funktionen, Module
Hersteller / Verbraucher	ein Teil erzeugt, was der andere verwendet	Module, Funktionen
einziges Datum	Kapselung einer Datenstruktur, Zusammenfassung der Operationen	Module, Funktionen
einige Operation	Operation, die nicht mehr zerlegbar ist	Funktionen

Stufen der Kopplung (von schlecht nach gut)

Stufe	zwischen Funktionen/Modulen	betrifft
Einbruch	Veränderung des Codes	Funktionen
volle Öffnung	Zugriff auf alle Daten, z.B. auf alle Attribute einer Klasse	(Module) Funktionen

Stufen der Kopplung (von schlecht nach gut)

Stufe	zwischen Funktionen/Modulen	betrifft
Einbruch	Veränderung des Codes	Funktionen
volle Öffnung	Zugriff auf alle Daten, z.B. auf alle Attribute einer Klasse	(Module) Funktionen
selektive Öffnung	bestimmte Attribute sind zugänglich oder global durch expliziten Export/Import	Module, Funktionen

Stufen der Kopplung (von schlecht nach gut)

Stufe	zwischen Funktionen/Modulen	betrifft
Einbruch	Veränderung des Codes	Funktionen
volle Öffnung	Zugriff auf alle Daten, z.B. auf alle Attribute einer Klasse	(Module) Funktionen
selektive Öffnung	bestimmte Attribute sind zugänglich oder global durch expliziten Export/Import	Module, Funktionen
Funktions-kopplung	nur Funktionsaufruf und Parameter	Module (Funktion)
keine Kopplung	Es besteht keine logische Beziehung (Zugriff ist syntaktisch unmöglich)	Module

Kriterien für einen guten Software-Entwurf

Jeder Entwurf ist ein Kompromiss.

- Geringe Kopplung zwischen allen Modulen
- Hoher Zusammenhalt in allen Funktionen und Modulen
- Kriterium der naiven Suche: Jede Einheit sollte einen leicht erkennbaren Sinn haben.

Kriterien für einen guten Software-Entwurf

Jeder Entwurf ist ein Kompromiss.

- Geringe Kopplung zwischen allen Modulen
- Hoher Zusammenhalt in allen Funktionen und Modulen
- Kriterium der naiven Suche: Jede Einheit sollte einen leicht erkennbaren Sinn haben.
- Abstraktion: Implementierungsdetails verbergen
- Kapselung von Dingen, die sich ändern können (Geheimnisprinzip; Information Hiding)

Kriterien für einen guten Software-Entwurf

Jeder Entwurf ist ein Kompromiss.

- Geringe Kopplung zwischen allen Modulen
- Hoher Zusammenhalt in allen Funktionen und Modulen
- Kriterium der naiven Suche: Jede Einheit sollte einen leicht erkennbaren Sinn haben.
- Abstraktion: Implementierungsdetails verbergen
- Kapselung von Dingen, die sich ändern können (Geheimnisprinzip; Information Hiding)
- Etwa gleich große Einheiten (Module, Funktionen). Abweichungen sollten plausibel sein; generell dürfen einfache Objekte größer sein als komplizierte.

Kriterien für einen guten Software-Entwurf (Forts.)

- Uniforme Entwurfsstrategie; wenn z.B. eine Schichtenstruktur gewählt wurde, sollte sie nicht an irgendeiner Stelle korrumpiert sein.
- Uniforme Gestaltung der Schnittstellen.
- Uniforme Benennung.

¹Nein, nicht *der* Michael Jackson.

Kriterien für einen guten Software-Entwurf (Forts.)

- Uniforme Entwurfsstrategie; wenn z.B. eine Schichtenstruktur gewählt wurde, sollte sie nicht an irgendeiner Stelle korrumpiert sein.
- Uniforme Gestaltung der Schnittstellen.
- Uniforme Benennung.
- Prinzip der Isomorphie zur Realität (Michael Jackson¹): Die Software sollte der Realität strukturell gleichen, damit die Änderungen der Realität in der Software leicht nachvollzogen werden können.

¹Nein, nicht *der* Michael Jackson.

Fragen



Wie kann man eine Software-Architektur bewerten?

Bewertung von Modularisierung

Software Architecture Analysis Method (SAAM) (Kazman u. a. 1996):

- ① Lege wichtige Qualitätsaspekte fest
- ② Beschreibe alternative Modularisierungen
- ③ Entwickle Szenarien für die Qualitätsaspekte
- ④ Spiele die Szenarien durch und bewerte die Modularisierungen
- ⑤ Betrachte Wechselwirkungen zwischen den Qualitätsaspekten
- ⑥ Fasse die Evaluation zusammen

Beispiel: Key Word in Context (KWIC) (Parnas 1972)

westfälischer Friede

Friede von Osnabrück

Bistum Osnabrück

Beispiel: Key Word in Context (KWIC) (Parnas 1972)

zyklische Vertauschung
jeder Zeile



westfälischer Friede	westfälischer Friede
Friede von Osnabrück	Friede westfälischer
Bistum Osnabrück	Friede von Osnabrück
	Osnabrück Friede von
	von Osnabrück Friede
	Bistum Osnabrück
	Osnabrück Bistum

Beispiel: Key Word in Context (KWIC) (Parnas 1972)

zyklische Vertauschung
jeder Zeile



westfälischer Friede
Friede von Osnabrück
Bistum Osnabrück

Sortierung der
Zeilen



westfälischer Friede
Friede westfälischer
Friede von Osnabrück
Osnabrück Friede von
von Osnabrück Friede

Bistum Osnabrück
Friede von Osnabrück
Friede westfälischer
Osnabrück Bistum
Osnabrück Friede von

von Osnabrück Friede
westfälischer Friede

Betrachtete Qualitätsaspekte und Szenarien

- Änderbarkeit
 - ▶ EingabefORMAT ändert sich
 - ▶ größere Dateien müssen verarbeitet werden
 - ▶ riesige Dateien müssen verarbeitet werden
- separate Entwickelbarkeit
 - ▶ verteile Arbeit an Entwicklungsteams
- Performanz
 - ▶ berechne KWIC für FB3-Webseiten

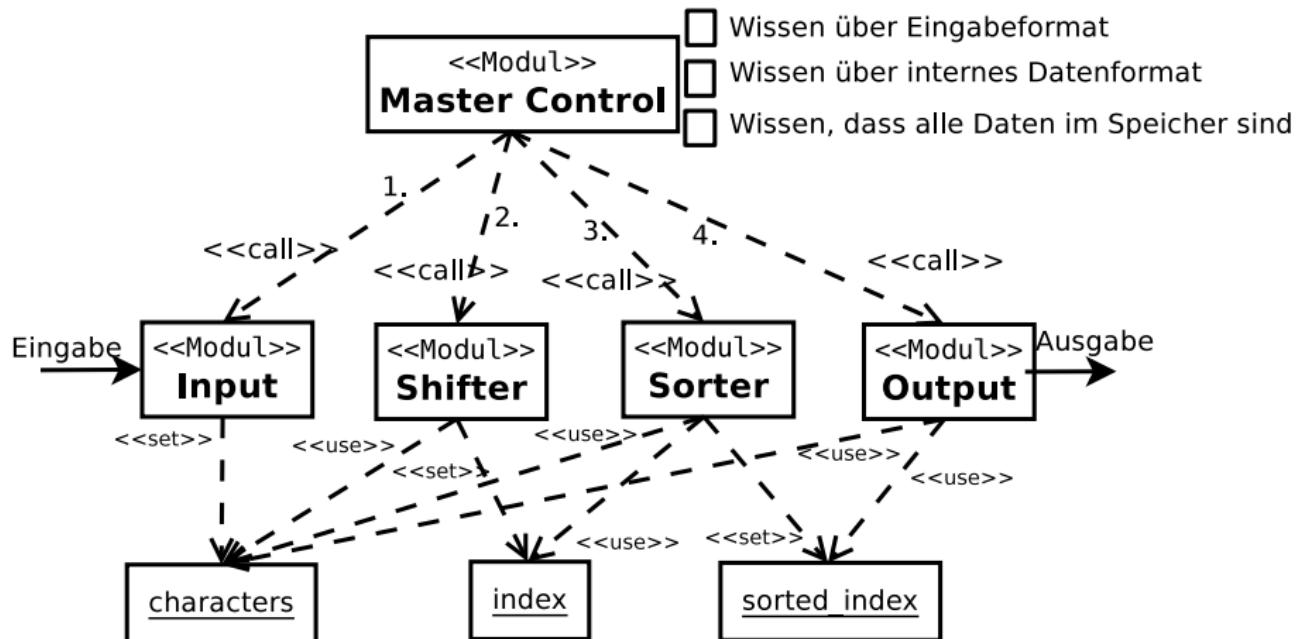
Ablauf

- ① Eingabe der Daten
- ② Interne Speicherung der Daten
- ③ Indizierung (Zeile, Wortanfang, Wortende)

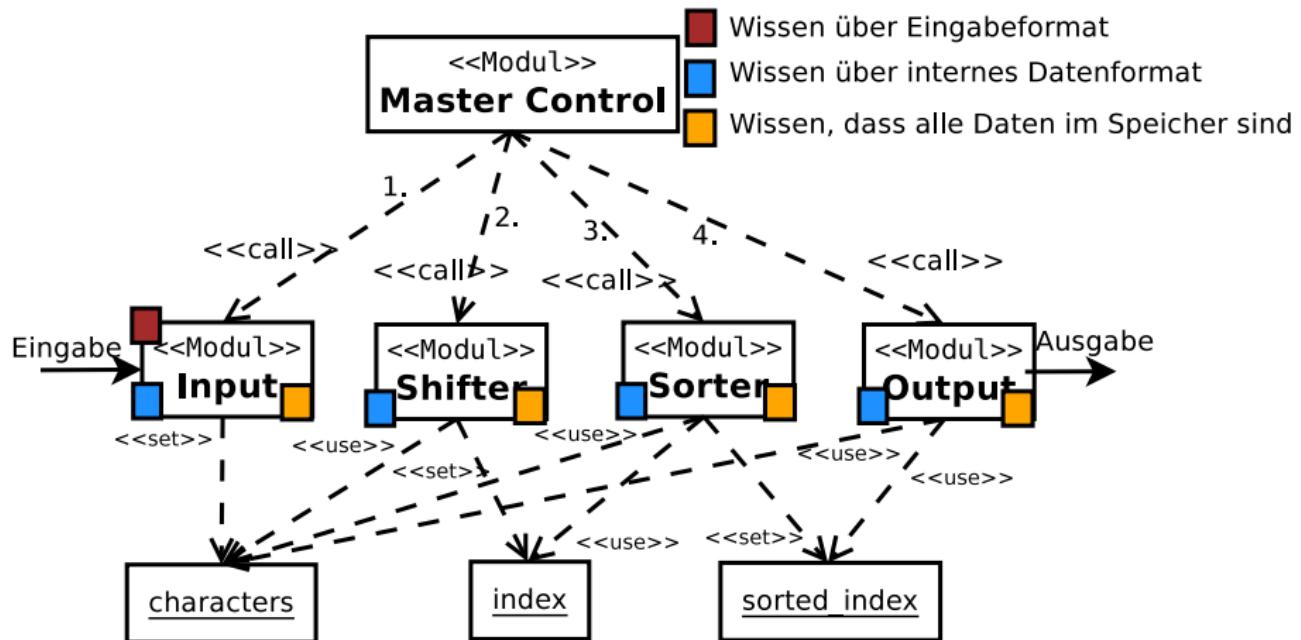
W e s t f ä l i s c h e r F r i e d e	(1, 1, 13) (1, 15, 20)
B i s t u m O s n a b r ü c k	(2, 1, 6) (2, 8, 16)
F r i e d e v o n O s n a b r ü c k	(3, 1, 6) (3, 8, 10) (3, 12, 20)

- ④ Zyklische Rotation der Indizierung
- ⑤ Sortierung der Indizierung
- ⑥ Ausgabe der Indizierung

Erste Modularisierung (ablauforientiert)



Erste Modularisierung (ablauforientiert)



Qualitäten

Änderbarkeit:

	betroffene Module			
	Input	Shifter	Sorter	Output
Eingabeformat	×			
größere Dateien	×	×	×	×
riesige Dateien	×	×	×	×

Getrennte Entwicklung:

- alle Datenstrukturen müssen bekannt sein
- komplexe Beschreibung notwendig

Performanz:

- schneller Zugriff auf globale Variablen

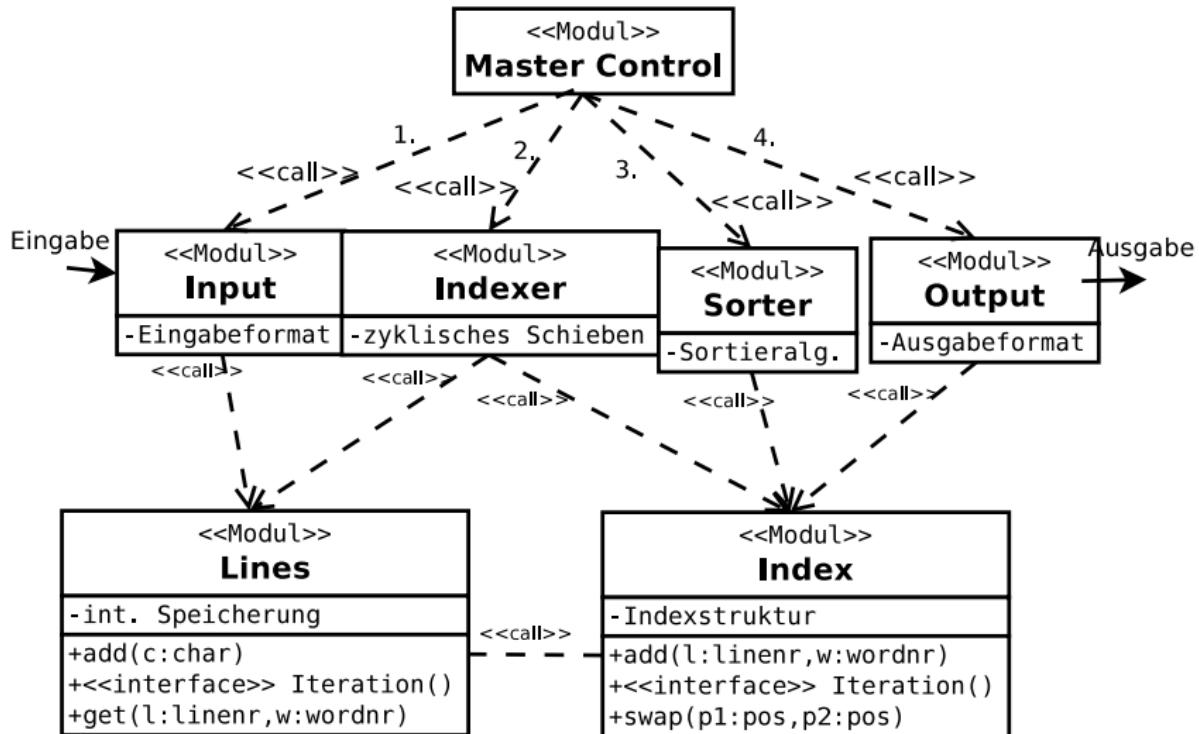
Geheimnisprinzip (Information Hiding) (Parnas 1972)

Definition

Geheimnisprinzip: Module verbergen alle Implementierungsentscheidungen, die sich ändern können, hinter einer abstrakten Schnittstelle.

Zweite Modularisierung (objektbasiert)

... nach dem Geheimnisprinzip (Information Hiding) (Parnas 1972)



Qualitäten

Änderbarkeit:

	betroffene Module					
	Input	Lines	Index	Indexer	Sorter	Output
Eingabeformat	x					
größere Dateien		x				
riesige Dateien		x				

Getrennte Entwicklung:

- nur Schnittstellen müssen bekannt sein

Performanz:

- zusätzliche Aufrufe

Abschließendes Wort zur Modularisierung

Es gibt viele Modularisierungsmöglichkeiten.

Jede ist gut für einen Zweck, schlecht für einen anderen.

Mit gängigen Programmiersprachen muss man sich auf eine festlegen.

Abschließendes Wort zur Modularisierung

Es gibt viele Modularisierungsmöglichkeiten.

Jede ist gut für einen Zweck, schlecht für einen anderen.

Mit gängigen Programmiersprachen muss man sich auf eine festlegen.

Definition

Querschnittsbelange (Cross-Cutting Concerns):

Implementierungsaspekte, die eine große Zahl von Modulen betreffen.

Beispiele: Fehlerbehandlung, Logging-Mechanismen, Vermeidung von Speicherlöchern etc.

Abschließendes Wort zur Modularisierung

Es gibt viele Modularisierungsmöglichkeiten.

Jede ist gut für einen Zweck, schlecht für einen anderen.

Mit gängigen Programmiersprachen muss man sich auf eine festlegen.

Definition

Querschnittsbelange (Cross-Cutting Concerns):

Implementierungsaspekte, die eine große Zahl von Modulen betreffen.

Beispiele: Fehlerbehandlung, Logging-Mechanismen, Vermeidung von Speicherlöchern etc.

Aspektorientierte Programmiersprachen erlauben eine separate Beschreibung dieser Aspekte und ein „Einweben“ des Aspekts in das System.